```
RRRRRRRRRRR      UUU          UUU  NNN         NNN     000000000     FFFFFFFFFFFFFFF  FFFFFFFFFFFFFFF
RRRRRRRRRRRR     UUU          UUU  NNN         NNN     000000000     FFFFFFFFFFFFFFF  FFFFFFFFFFFFFFF
RRRRRRRRRRRR     UUU          UUU  NNN         NNN     000000000     FFFFFFFFFFFFFFF  FFFFFFFFFFFFFFF
RRR          RRR UUU          UUU  NNN         NNN  000         000  FFF              FFF
RRR          RRR UUU          UUU  NNN         NNN  000         000  FFF              FFF
RRR          RRR UUU          UUU  NNN         NNN  000         000  FFF              FFF
RRR          RRR UUU          UUU  NNNNNN      NNN  000         000  FFF              FFF
RRR          RRR UUU          UUU  NNNNNN      NNN  000         000  FFF              FFF
RRR          RRR UUU          UUU  NNNNNN      NNN  000         000  FFF              FFF
RRRRRRRRRRRR     UUU          UUU  NNN  NNN    NNN  000         000  FFFFFFFFFFFF     FFFFFFFFFFFF
RRRRRRRRRRRR     UUU          UUU  NNN  NNN    NNN  000         000  FFFFFFFFFFFF     FFFFFFFFFFFF
RRRRRRRRRRRR     UUU          UUU  NNN  NNN    NNN  000         000  FFFFFFFFFFFF     FFFFFFFFFFFF
RRR    RRR       UUU          UUU  NNN      NNNNNN  000         000  FFF              FFF
RRR    RRR       UUU          UUU  NNN      NNNNNN  000         000  FFF              FFF
RRR    RRR       UUU          UUU  NNN      NNNNNN  000         000  FFF              FFF
RRR       RRR    UUU          UUU  NNN         NNN  000         000  FFF              FFF
RRR       RRR    UUU          UUU  NNN         NNN  000         000  FFF              FFF
RRR       RRR    UUU          UUU  NNN         NNN  000         000  FFF              FFF
RRR          RRR UUUUUUUUUUUUUUU  NNN         NNN     000000000     FFF              FFF
RRR          RRR UUUUUUUUUUUUUUU  NNN         NNN     000000000     FFF              FFF
RRR          RRR UUUUUUUUUUUUUUU  NNN         NNN     000000000     FFF              FFF
RRR          RRR UUUUUUUUUUUUUUU  NNN         NNN     000000000     FFF              FFF
RRR          RRR UUUUUUUUUUUUUUU  NNN         NNN     000000000     FFF              FFF
```

_$2

```
Sym
---
NDX
NDX
NUM
NUM
OPE
OUT
PAC
PAC
PAC
PAC
PAC
PAC
PAC
PAD
PAG
PAG
PAG
PAG
PAG
PAG
PAG
PER
PUT
RCO
RIN

RLI
RNO
RNO
RTY
SAV
STR
STR
STR
STR
STR
STR
STR

 STR
STR
STR
STR
STR
STR
STR
STR
STR
STR
STR
STR
STR
STR
STR
STR
```

```
NN      NN  DDDDDDDD    XX      XX  VV      VV  MM      MM    SSSSSSSS
NN      NN  DDDDDDDD    XX      XX  VV      VV  MM      MM    SSSSSSSS
NN      NN  DD      DD  XX      XX  VV      VV  MMMM  MMMM  SS
NN      NN  DD      DD  XX      XX  VV      VV  MMMM  MMMM  SS
NNNN    NN  DD      DD    XX  XX    VV      VV  MM  MM  MM  SS
NNNN    NN  DD      DD    XX  XX    VV      VV  MM  MM  MM  SS
NN  NN  NN  DD      DD      XX      VV      VV  MM      MM    SSSSSS
NN  NN  NN  DD      DD      XX      VV      VV  MM      MM    SSSSSS
NN    NNNN  DD      DD    XX  XX    VV      VV  MM      MM          SS
NN    NNNN  DD      DD    XX  XX    VV      VV  MM      MM          SS
NN      NN  DD      DD  XX      XX    VV  VV    MM      MM          SS      ....
NN      NN  DD      DD  XX      XX    VV  VV    MM      MM          SS      ....
NN      NN  DDDDDDDD    XX      XX      VV      MM      MM  SSSSSSSS        ....
NN      NN  DDDDDDDD    XX      XX      VV      MM      MM  SSSSSSSS        ....

LL            IIIIII      SSSSSSSS
LL            IIIIII      SSSSSSSS
LL              II      SS
LL              II      SS
LL              II      SS
LL              II      SS
LL              II        SSSSSS
LL              II        SSSSSS
LL              II              SS
LL              II              SS
LL              II              SS
LL              II              SS
LLLLLLLLLL    IIIIII    SSSSSSSS
LLLLLLLLLL    IIIIII    SSSSSSSS
```

```
;  1          0001  0 %TITLE 'NDXVMS -- DSRINDEX/INDEX Command line interface'
;  2          0002  0 MODULE NDXVMS   (IDENT = 'V04-000', LANGUAGE (BLISS32),
;  3          0003  0                 ADDRESSING_MODE (EXTERNAL    = LONG_RELATIVE,
;  4          0004  0                                  NONEXTERNAL = LONG_RELATIVE)
;  5          0005  0                 ) =
;  6          0006  0
;  7          0007  1 BEGIN
;  8          0008  1
;  9          0009  1 !
; 10          0010  1 !*****************************************************************************
; 11          0011  1 !*                                                                           *
; 12          0012  1 !*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                  *
; 13          0013  1 !*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                    *
; 14          0014  1 !*  ALL RIGHTS RESERVED.                                                     *
; 15          0015  1 !*                                                                           *
; 16          0016  1 !*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
; 17          0017  1 !*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE  *
; 18          0018  1 !*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
; 19          0019  1 !*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
; 20          0020  1 !*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
; 21          0021  1 !*  TRANSFERRED.                                                             *
; 22          0022  1 !*                                                                           *
; 23          0023  1 !*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
; 24          0024  1 !*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
; 25          0025  1 !*  CORPORATION.                                                             *
; 26          0026  1 !*                                                                           *
; 27          0027  1 !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
; 28          0028  1 !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
; 29          0029  1 !*                                                                           *
; 30          0030  1 !*                                                                           *
; 31          0031  1 !*****************************************************************************
; 32          0032  1 !
; 33          0033  1
; 34          0034  1 !++
; 35          0035  1 ! FACILITY:
; 36          0036  1 !   DSR (Digital Standard RUNOFF) /DSRPLUS DSRINDEX/INDEX Utility
; 37          0037  1 !
; 38          0038  1 ! ABSTRACT:
; 39          0039  1 !
; 40          0040  1 !       This module is the INDEX command line interface module.
; 41          0041  1 !
; 42          0042  1 !       Much of the code to parse and validate qualifier
; 43          0043  1 !       values may be removed when the VMS CLI interface routines
; 44          0044  1 !       implement value validation.
; 45          0045  1 !
; 46          0046  1 ! ENVIRONMENT:   VAX/VMS User Mode
; 47          0047  1 !
; 48          0048  1 ! AUTHOR:        JPK
; 49          0049  1 !
; 50          0050  1 ! CREATION DATE: February-1982
; 51          0051  1 !
; 52          0052  1 ! MODIFIED BY:
; 53          0053  1 !
; 54          0054  1 !       012     JPK00023        20-May-1983
; 55          0055  1 !               Modified INDEX, NDXT20 and NDXVMS to check status of
; 56          0056  1 !               $XPO_PARSE_SPEC to avoid error messages from XPORT.
; 57          0057  1 !
```

```
;    58          0058  1 !          011     JPK00022        30-Mar-1983
;    59          0059  1 !                  Modified NDXVMS, NDXFMT, NDXPAG, NDXVMSMSG and NDXVMSREQ
;    60          0060  1 !                  to generate TEX output. Added module NDXTEX.
;    61          0061  1 !
;    62          0062  1 !          010     JPK00019        14-MAR-1983
;    63          0063  1 !                  Modified NDXVMS to conditionalize /PAGE_NUMBERS=[NO]MERGE
;    64          0064  1 !                  and /PAGE_NUMBERS=STANDARD for DSRPLUS only.
;    65          0065  1 !
;    66          0066  1 !          009     JPK00016        23-Feb-1983
;    67          0067  1 !                  Modified NDXVMS to change the default number of lines per page
;    68          0068  1 !                  when /TELLTALE is specified but /LINES is not.
;    69          0069  1 !
;    70          0070  1 !          008     JPK00015        04-Feb-1983
;    71          0071  1 !                  Cleaned up module names, modified revision history to
;    72          0072  1 !                  conform with established standards. Updated copyright dates.
;    73          0073  1 !
;    74          0074  1 !          007     JPK00013        31-Jan-1983
;    75          0075  1 !                  Changed default subindex level value from 6 to 99 in NDXVMS
;    76          0076  1 !                  and NDXCLIDMP. This value is the subindexing level.
;    77          0077  1 !                  It is NOT A HEADER LEVEL.
;    78          0078  1 !
;    79          0079  1 !          006     JPK00012        24-Jan-1983
;    80          0080  1 !                  Modified NDXVMSMSG.MSG to define error messages for both
;    81          0081  1 !                  DSRINDEX and INDEX.
;    82          0082  1 !                  Added require of NDXVMSREQ.R32 to NDXOUT, NDXFMT, NDXDAT,
;    83          0083  1 !                  INDEX, NDXMSG, NDXXTN, NDXTMS, NDXVMS and NDXPAG for BLISS32.
;    84          0084  1 !                  Since this file defines the error message literals,
;    85          0085  1 !                  the EXTERNAL REFERENCEs for the error message literals
;    86          0086  1 !                  have been removed.
;    87          0087  1 !
;    88          0088  1 !          005     JPK00011        24-Jan-1983
;    89          0089  1 !                  Changed CMDBLK [NDX$G_LEVEL] to CMDBLK [NDX$H_LEVEL]
;    90          0090  1 !                  Changed CMDBLK [NDX$H_FORMAT] to CMDBLK [NDX$H_LAYOUT]
;    91          0091  1 !                  Changed CMDBLK [NDX$V_TMS11] and CMDBLK [NDX$V_TEX] to CMDBLK [NDX$H_FORMAT]
;    92          0092  1 !                  Changed comparisons of (.CHRSIZ EQLA CHRSZA) to
;    93          0093  1 !                  (.CMDBLK [NDX$H_FORMAT] EQL TMS11_A).
;    94          0094  1 !                  Definitions were changed in NDXCLI and references to the
;    95          0095  1 !                  effected fields were changed in NDXPAG, NDXFMT, INDEX, NDXVMS
;    96          0096  1 !                  and NDXCLIDMP.
;    97          0097  1 !
;    98          0098  1 !          004     RER00002        20-Jan-1983
;    99          0099  1 !                  Modified VMS command line interface module NDXVMS:
;   100          0100  1 !                    - changed /FORMAT qualifier to /LAYOUT.
;   101          0101  1 !                    - changed use of /RESERVE and /REQUIRE for DSRPLUS.
;   102          0102  1 !                    - added code for new DSRPLUS qualifiers /FORMAT and
;   103          0103  1 !                      /TELLTALE_HEADINGS.
;   104          0104  1 !                  Added fields to NDXCLI for new qualifiers: NDX$V_TELLTALE
;   105          0105  1 !                  and NDX$V_TEX.
;   106          0106  1 !                  Conditionalized output of NDX$V_PAGE_MERGE in NDXCLIDMP to
;   107          0107  1 !                  account for different DSR and DSRPLUS default values.
;   108          0108  1 !
;   109          0109  1 !          003     RER00001        17-Dec-1982
;   110          0110  1 !                  Modified VMS command line interface module NDXVMS:
;   111          0111  1 !                    - Added code to treat keyword NORUNNING in same way as
;   112          0112  1 !                      keyword STANDARD.
;   113          0113  1 !                    - Added code for new DSR qualifiers /RESERVE and /REQUIRE.
;   114          0114  1 !                    - Changed header level default value from 99 to 6.
```

```
;  115     0115  1 !                   - Conditionalized code to compile for DSRPLUS if BLISS
;  116     0116  1 !                     /VARIANT = 8192 is used; otherwise, to compile for DSR.
;  117     0117  1 !                   - Deleted foreign-command code; INDEX is now called
;  118     0118  1 !                     as a subcommand of DSR.
;  119     0119  1 !
;  120     0120  1 !           002     JPK00001        13-Aug-1982
;  121     0121  1 !                   Removed reference to CLI$END_PARSE in NDXVMS. It is no longer
;  122     0122  1 !                   supported by VMS.
;  123     0123  1 !
;  124     0124  1 !--
;  125     0125  1
;  126     0126  1 !
;  127     0127  1 ! INCLUDE FILES:
;  128     0128  1 !
;  129     0129  1 LIBRARY 'SYS$LIBRARY:STARLET.L32';      ! System macro library
;  130     0130  1
;  131     0131  1 LIBRARY 'SYS$LIBRARY:TPAMAC.L32';       ! TPARSE macros
;  132     0132  1
;  133     0133  1 LIBRARY 'SYS$LIBRARY:XPORT';            ! Transportable BLISS library
;  134     0134  1
;  135     0135  1 SWITCHES LIST (REQUIRE);                ! Print require files
;  136     0136  1
;  137     0137  1 REQUIRE 'REQ:NDXCLI';                   ! Command line information block
```

```
;  R0138  1        !                                IDENT = 0V04-00004
;  R0139  1        !
;  R0140  1        !*****************************************************************************
;  R0141  1        !*                                                                           *
;  R0142  1        !*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                  *
;  R0143  1        !*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                   *
;  R0144  1        !*  ALL RIGHTS RESERVED.                                                     *
;  R0145  1        !*                                                                           *
;  R0146  1        !*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED    *
;  R0147  1        !*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
;  R0148  1        !*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER    *
;  R0149  1        !*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY    *
;  R0150  1        !*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY    *
;  R0151  1        !*  TRANSFERRED.                                                             *
;  R0152  1        !*                                                                           *
;  R0153  1        !*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE    *
;  R0154  1        !*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT    *
;  R0155  1        !*  CORPORATION.                                                             *
;  R0156  1        !*                                                                           *
;  R0157  1        !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS    *
;  R0158  1        !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
;  R0159  1        !*                                                                           *
;  R0160  1        !*                                                                           *
;  R0161  1        !*****************************************************************************
;  R0162  1        !
;  R0163  1
;  R0164  1        !++
;  R0165  1        ! FACILITY:
;  R0166  1        !   DSR (Digital Standard RUNOFF) /DSRPLUS DSRINDEX/INDEX Utility
;  R0167  1        !
;  R0168  1        ! ABSTRACT:       INDEX command line definitions
;  R0169  1        !
;  R0170  1        ! ENVIRONMENT:    Transportable
;  R0171  1        !
;  R0172  1        ! AUTHOR:         JPK
;  R0173  1        !
;  R0174  1        ! CREATION DATE: January 1982
;  R0175  1        !
;  R0176  1        ! MODIFIED BY:
;  R0177  1        !
;  R0178  1        !       004     JPK00015        04-Feb-1983
;  R0179  1        !               Cleaned up module names, modified revision history to
;  R0180  1        !               conform with established standards. Updated copyright dates.
;  R0181  1        !
;  R0182  1        !       003     JPK00011        24-Jan-1983
;  R0183  1        !               Changed CMDBLK [NDX$G_LEVEL] to CMDBLK [NDX$H_LEVEL]
;  R0184  1        !               Changed CMDBLK [NDX$H_FORMAT] to CMDBLK [NDX$H_LAYOUT]
;  R0185  1        !               Changed CMDBLK [NDX$V_TMS11] and CMDBLK [NDX$V_TEX] to CMDBLK [NDX$H_FORMAT]
;  R0186  1        !               Changed comparisons of (.CHRSIZ EQLA CHRSZA) to
;  R0187  1        !               (.CMDBLK [NDX$H_FORMAT] EQL TMS11_A).
;  R0188  1        !               Definitions were changed in NDXCLI and references to the
;  R0189  1        !               effected fields were changed in NDXPAG, NDXFMT, INDEX, NDXVMS
;  R0190  1        !               and NDXCLIDMP.
;  R0191  1        !
;  R0192  1        !       002     RER00002        20-Jan-1983
;  R0193  1        !               Modified VMS command line interface module NDXVMS:
;  R0194  1        !                 - changed /FORMAT qualifier to /LAYOUT.
```

```
;  R0195  1     !                 - changed use of /RESERVE and /REQUIRE for DSRPLUS.
;  R0196  1     !                 - added code for new DSRPLUS qualifiers /FORMAT and
;  R0197  1     !                   /TELLTALE_HEADINGS.
;  R0198  1     !               Added fields to NDXCLI for new qualifiers: NDX$V_TELLTALE
;  R0199  1     !               and NDX$V_TEX.
;  R0200  1     !               Conditionalized output of NDX$V_PAGE_MERGE in NDXCLIDMP to
;  R0201  1     !               account for different DSR and DSRPLUS default values.
;  R0202  1     !
;  R0203  1     !--
;  R0204  1
```

```
;  R0205  1     !
;  R0206  1     !  NDXCMD_FIELDS
;  R0207  1     !
;  R0208  1     $FIELD ndxcmd_fields =
;  R0209  1         SET
;  R0210  1
;  R0211  1         NDX$V_OPTIONS       = [$INTEGER],          ! Command option indicators:
;  R0212  1
;  R0213  1             $OVERLAY (NDX$V_OPTIONS)
;  R0214  1
;  R0215  1             NDX$V_INPUT_CONCAT      = [$BIT],      ! Input file concatenated to previous
;  R0216  1             NDX$V_OUTPUT            = [$BIT],      ! Generate output file
;  R0217  1             NDX$V_REQUIRE           = [$BIT],      ! Require file specified
;  R0218  1             NDX$V_PAGES             = [$BIT],      ! Include page references in index
;  R0219  1             NDX$V_OVERRIDE          = [$BIT],      ! Override master index information
;  R0220  1             NDX$V_STANDARD_PAGE     = [$BIT],      ! Generate standard page numbers
;  R0221  1             NDX$V_CONTINUATION      = [$BIT],      ! Generate continuation headings
;  R0222  1             NDX$V_GUIDE             = [$BIT],      ! Generate guide headings
;  R0223  1             NDX$V_WORD_SORT         = [$BIT],      ! Sort entries word by word
;  R0224  1             NDX$V_LOG               = [$BIT],      ! Generate /LOG message
;  R0225  1             NDX$V_MASTER            = [$BIT],      ! Generate a master index
;  R0226  1             NDX$V_PAGE_MERGE        = [$BIT],      ! Merge adjacent page references
;  R0227  1             NDX$V_TELLTALE          = [$BIT],      ! Generate telltale headings
;  R0228  1
;  R0229  1             $CONTINUE
;  R0230  1
;  R0231  1         NDX$H_FORMAT        = [$SHORT_INTEGER],    ! Output format: DSR, TMS, TEX
;  R0232  1         NDX$H_LAYOUT        = [$SHORT_INTEGER],    ! Output layout type
;  R0233  1         NDX$H_NONALPHA      = [$SHORT_INTEGER],    ! Treatment of leading nonalphas during sort
;  R0234  1         NDX$H_LEVEL         = [$SHORT_INTEGER],    ! Deepest level to include in index
;  R0235  1         NDX$G_COLUMN_WID    = [$INTEGER],          ! Column width
;  R0236  1         NDX$G_GUTTER_WID    = [$INTEGER],          ! Gutter width
;  R0237  1         NDX$G_LINES_PAGE    = [$INTEGER],          ! Lines per page
;  R0238  1         NDX$G_RESERVE_LINES = [$INTEGER],          ! Number of lines to reserve when requiring a file
;  R0239  1         NDX$G_SEPARATE_WIDTH= [$INTEGER],          ! Width of reference portion of entry
;  R0240  1         NDX$T_MASTER_BOOK   = [$DESCRIPTOR(DYNAMIC)], ! Book name descriptor for Master indexing
;  R0241  1         NDX$T_INPUT_FILE    = [$DESCRIPTOR(DYNAMIC)], ! Input file name descriptor
;  R0242  1         NDX$T_OUTPUT_FILE   = [$DESCRIPTOR(DYNAMIC)], ! Output file name descriptor
;  R0243  1         NDX$T_REQUIRE_FILE  = [$DESCRIPTOR(DYNAMIC)], ! Require file name descriptor
;  R0244  1         NDX$T_RELATED_FILE  = [$DESCRIPTOR(DYNAMIC)], ! Related file name descriptor is saved here
;  R0245  1                                                       !  by NDXINP for later use by MAKNDX
;  R0246  1         NDX$T_COMMAND_LINE  = [$DESCRIPTOR(DYNAMIC)]  ! Copy of entire command line
;  R0247  1
;  R0248  1         TES;
;  R0249  1     !
;  R0250  1     !  End of NDXCMD_FIELDS
;  R0251  1     !
;  R0252  1
;  R0253  1     LITERAL
;  R0254  1         NDXCMD$K_LENGTH = $FIELD_SET_SIZE;
;  R0255  1
;  R0256  1     MACRO
;  R0257  1         $NDXCMD = BLOCK [NDXCMD$K_LENGTH] FIELD (NDXCMD_FIELDS) %;
;  R0258  1
;  R0259  1     $LITERAL                                       ! Output formats (NDX$H_FORMAT)
;  R0260  1         DSR                 = $DISTINCT,          ! Runoff
;  R0261  1         TMS11_A             = $DISTINCT,          ! TMS=A
```

```
;  R0262  1          TMS11_E                 = $DISTINCT,    ! TMS=E
;  R0263  1          TEX                     = $DISTINCT;    ! TEX
;  R0264  1
;  R0265  1      $LITERAL                                    ! Output layouts (NDX$H_LAYOUT)
;  R0266  1          TWO_COLUMN              = $DISTINCT,    ! Normal two column format
;  R0267  1          ONE_COLUMN              = $DISTINCT,    ! Normal one column format
;  R0268  1          SEPARATE                = $DISTINCT,    ! Separate reference format
;  R0269  1          GALLEY                  = $DISTINCT;    ! TMS11 Galley format
;  R0270  1
;  R0271  1      $LITERAL                                    ! Treatment of leading nonalphas during sort (NDX$H_NONALPHA)
;  R0272  1          BEFORE                  = $DISTINCT,    ! Leading nonalphas sort before alphas
;  R0273  1          AFTER                   = $DISTINCT,    ! Leading nonalphas sort after alphas
;  R0274  1          IGNORE                  = $DISTINCT;    ! Leading nonalphas are ignored
;  R0275  1
;  R0276  1      !
;  R0277  1      !--     End of NDXCLI.REQ
```

```
;  138     0278  1
;  139     0279  1 REQUIRE 'REQ:NDXVMSREQ';                    ! Error message definitions
```

```
;  R0280  1        !
;  R0281  1        ! Version:      'V04-000'
;  R0282  1        !
;  R0283  1        !****************************************************************************
;  R0284  1        !*                                                                          *
;  R0285  1        !*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
;  R0286  1        !*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                  *
;  R0287  1        !*  ALL RIGHTS RESERVED.                                                    *
;  R0288  1        !*                                                                          *
;  R0289  1        !*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
;  R0290  1        !*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE  *
;  R0291  1        !*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
;  R0292  1        !*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
;  R0293  1        !*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
;  R0294  1        !*  TRANSFERRED.                                                            *
;  R0295  1        !*                                                                          *
;  R0296  1        !*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
;  R0297  1        !*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
;  R0298  1        !*  CORPORATION.                                                            *
;  R0299  1        !*                                                                          *
;  R0300  1        !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
;  R0301  1        !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                 *
;  R0302  1        !*                                                                          *
;  R0303  1        !*                                                                          *
;  R0304  1        !****************************************************************************
;  R0305  1        !
;  R0306  1        !
;  R0307  1        !++
;  R0308  1        ! FACILITY:
;  R0309  1        !   DSR (Digital Standard RUNOFF) /DSRPLUS DSRINDEX/INDEX Utility
;  R0310  1        !
;  R0311  1        ! ABSTRACT:
;  R0312  1        !       This file contains external references to the error message numbers
;  R0313  1        !       for DSRINDEX/INDEX.
;  R0314  1        !
;  R0315  1        !       New messages must be defined in NDXVMSMSG.MSG and referenced here:
;  R0316  1        !       both in the MACRO section (for DSRINDEX) and the EXTERNAL LITERAL
;  R0317  1        !       section (for INDEX)
;  R0318  1        !
;  R0319  1        ! ENVIRONMENT:    VAX/VMS User Mode
;  R0320  1        !
;  R0321  1        ! AUTHOR:         JPK
;  R0322  1        !
;  R0323  1        ! CREATION DATE: 01-Feb-1983
;  R0324  1        !
;  R0325  1        ! MODIFIED BY:
;  R0326  1        !
;  R0327  1        !       004     JPK00022        30-Mar-1983
;  R0328  1        !               Modified NDXVMS, NDXFMT, NDXPAG, NDXVMSMSG and NDXVMSREQ
;  R0329  1        !               to generate TEX output. Added module NDXTEX.
;  R0330  1        !
;  R0331  1        !       003     JPK00021        28-Mar-1983
;  R0332  1        !               Modified NDXT20 to include E2.0 functionality.
;  R0333  1        !               Modified NDXCLIDMP, NDXFMT, NDXPAG, NDXVRS to require RNODEF
;  R0334  1        !               for BLISS36 and to remove any conditional require based on
;  R0335  1        !               DSRPLUS_DEF.
;  R0336  1        !
```

```
;  R0337  1      !        002     JPK00010        04-Feb-1983
;  R0338  1      !                Cleaned up module names, modified revision history to
;  R0339  1      !                conform with established standards. Updated copyright dates.
;  R0340  1      !
;  R0341  1      !--
;  R0342  1
;  R0343  1      REQUIRE 'REQ:RNODEF';
```

```
;  R0344  1        !
;  R0345  1        ! Version:      'V04-000'
;  R0346  1        !
;  R0347  1        !****************************************************************************
;  R0348  1        !*                                                                          *
;  R0349  1        !*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
;  R0350  1        !*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                  *
;  R0351  1        !*  ALL RIGHTS RESERVED.                                                    *
;  R0352  1        !*                                                                          *
;  R0353  1        !*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
;  R0354  1        !*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE  *
;  R0355  1        !*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
;  R0356  1        !*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
;  R0357  1        !*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
;  R0358  1        !*  TRANSFERRED.                                                            *
;  R0359  1        !*                                                                          *
;  R0360  1        !*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
;  R0361  1        !*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
;  R0362  1        !*  CORPORATION.                                                            *
;  R0363  1        !*                                                                          *
;  R0364  1        !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
;  R0365  1        !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                 *
;  R0366  1        !*                                                                          *
;  R0367  1        !*                                                                          *
;  R0368  1        !****************************************************************************
;  R0369  1        !
;  R0370  1
;  R0371  1        !++
;  R0372  1        ! FACILITY:      DSR (Digital Standard RUNOFF) / DSRPLUS
;  R0373  1        !
;  R0374  1        ! ABSTRACT:
;  R0375  1        !       Converts BLISS/VARIANT values into useful names.
;  R0376  1        !
;  R0377  1        ! ENVIRONMENT:   Transportable BLISS
;  R0378  1        !
;  R0379  1        ! AUTHOR:        Rich Friday
;  R0380  1        !
;  R0381  1        ! CREATION DATE: 1978
;  R0382  1        !
;  R0383  1        ! MODIFIED BY:
;  R0384  1        !
;  R0385  1        !       016     KAD00016        Ray Marshall    19-Mar-1984
;  R0386  1        !               Added GERMAN, FRENCH, & ITALIAN.
;  R0387  1        !
;  R0388  1        !       015     KAD00015        Keith Dawson    18-Apr-1983
;  R0389  1        !               Made the LN01 conditional the default for vanilla DSR --
;  R0390  1        !                 its value is 0 (no variant supplied).
;  R0391  1        !
;  R0392  1        !       014     KAD00014        Keith Dawson    22-Mar-1983
;  R0393  1        !               Asserted the LN01 conditional when DSRPLUS is asserted.
;  R0394  1        !
;  R0395  1        !       013     KAD00013        Keith Dawson    20-Mar-1983
;  R0396  1        !               Removed all references to .BIX and .BTC files.
;  R0397  1        !
;  R0398  1        !       012     KAD00012        Keith Dawson    07-Mar-1983
;  R0399  1        !               Global edit of all modules. Updated module names, idents,
;  R0400  1        !               copyright dates. Changed require files to BLISS library.
```

```
;  R0401  1     !
;  R0402  1     !--
;  R0403  1
;  R0404  1     !++
;  R0405  1     !       D E F I N I T I O N   O F  /VARIANT   B I T S
;  R0406  1     !
;  R0407  1     !       The bit assignments are as follows:
;  R0408  1     !
;  R0409  1     !       Bit  Weight   Meaning
;  R0410  1     !-------------------------------------------------------------------
;  R0411  1     !       --      0     If no /VARIANT is supplied (as for vanilla DSR),
;  R0412  1     !                     compile with LN01 support. LN01 support is also
;  R0413  1     !                     implied by the DSRPLUS variant.
;  R0414  1     !
;  R0415  1     !        0      1     CLEAR =   Unassigned
;  R0416  1     !                     SET   =   Unassigned
;  R0417  1     !
;  R0418  1     !        1      2     CLEAR =   Normal compile
;  R0419  1     !                     SET   =   Compile for DSRPLUS
;  R0420  1     !
;  R0421  1     !       4-6    16     CLEAR =   English (American) version
;  R0422  1     !                     SET   =   16 = German (Austrian)
;  R0423  1     !                               32 = French
;  R0424  1     !                               48 = Italian
;  R0425  1     !--
;  R0426  1
;  R0427  1     !-------------------------------------------------------------------
;  R0428  1     !   This variable (LN01) controls whether or not to compile an LN01-flavored
;  R0429  1     !   DSR. It is asserted by default, and also whenever DSRPLUS is asserted.
;  R0430  1     !
;  R0431  1     !   Modules utilizing LN01 are:
;  R0432  1     !
;  R0433  1     !       DOOPTS  NOUT
;  R0434  1
;  R0435  1     COMPILETIME
;  R0436  1         ln01 =
;  R0437  2             ( (%VARIANT EQL 0) OR  %VARIANT/2 )
;  R0438  1         ;
;  R0439  1
;  R0440  1     !-------------------------------------------------------------------
;  R0441  1     !   This variable (DSRPLUS) controls compilation for the DSRPLUS program.
;  R0442  1     !
;  R0443  1     !   All modules utilize DSRPLUS.
;  R0444  1
;  R0445  1     COMPILETIME
;  R0446  1         dsrplus =
;  R0447  2             ( %VARIANT/2 )
;  R0448  1         ;
;  R0449  1
;  R0450  1     !-------------------------------------------------------------------
;  R0451  1     !   This variable (FLIP) controls compilation of FLIP features of DSRPLUS.
;  R0452  1     !   It assures that FLIP features are compiled only on VMS systems.
;  R0453  1     !
;  R0454  1     !   Modules utilizing FLIP are many and various.
;  R0455  1
;  R0456  1     COMPILETIME
;  R0457  1         flip =
```

```
;  R0458  2              ( %VARIANT/2 AND %BLISS(BLISS32) )
;  R0459  1          ;
;  R0460  1
;  R0461  1      !-----------------------------------------------------------------------
;  R0462  1      !       4-6    16     CLEAR =   English (American) version
;  R0463  1      !                     SET   =   16 = German (Austrian)
;  R0464  1      !                               32 = French
;  R0465  1      !                               48 = Italian
;  R0466  1      COMPILETIME
;  R0467  1          German  = ( %VARIANT/16 AND NOT %VARIANT/32 AND NOT %VARIANT/64 ) ;
;  R0468  1      COMPILETIME
;  R0469  1          French  = ( NOT %VARIANT/16 AND %VARIANT/32 AND NOT %VARIANT/64 ) ;
;  R0470  1      COMPILETIME
;  R0471  1          Italian = ( %VARIANT/16 AND %VARIANT/32 AND NOT %VARIANT/64 ) ;
;  R0472  1      !-----------------------------------------------------------------------
;  R0473  1      !                       End of RNODEF.REQ
```

```
;   R0474  1
; LR0475  1     %IF NOT DSRPLUS
;   R0476  1     %THEN
;   R0477  1
;   R0478  1     MACRO
;   R0479  1          INDEX$_BADLOGIC  = DSRINDEX$_BADLOGIC  %,
;   R0480  1          INDEX$_BADVALUE  = DSRINDEX$_BADVALUE  %,
;   R0481  1          INDEX$_INSVIRMEM = DSRINDEX$_INSVIRMEM %,
;   R0482  1          INDEX$_LINELENG  = DSRINDEX$_LINELENG  %,
;   R0483  1          INDEX$_NOREF     = DSRINDEX$_NOREF     %,
;   R0484  1          INDEX$_OPENIN    = DSRINDEX$_OPENIN    %,
;   R0485  1          INDEX$_OPENOUT   = DSRINDEX$_OPENOUT   %,
;   R0486  1          INDEX$_TOOMANY   = DSRINDEX$_TOOMANY   %,
;   R0487  1          INDEX$_VALERR    = DSRINDEX$_VALERR    %,
;   R0488  1          INDEX$_CANTBAL   = DSRINDEX$_CANTBAL   %,
;   R0489  1          INDEX$_CLOSEQUOT = DSRINDEX$_CLOSEQUOT %,
;   R0490  1          INDEX$_CONFQUAL  = DSRINDEX$_CONFQUAL  %,
;   R0491  1          INDEX$_CTRLCHAR  = DSRINDEX$_CTRLCHAR  %,
;   R0492  1          INDEX$_DOESNTFIT = DSRINDEX$_DOESNTFIT %,
;   R0493  1          INDEX$_DUPBEGIN  = DSRINDEX$_DUPBEGIN  %,
;   R0494  1          INDEX$_EMPTYIN   = DSRINDEX$_EMPTYIN   %,
;   R0495  1          INDEX$_IGNORED   = DSRINDEX$_IGNORED   %,
;   R0496  1          INDEX$_INVINPUT  = DSRINDEX$_INVINPUT  %,
;   R0497  1          INDEX$_INVRECORD = DSRINDEX$_INVRECORD %,
;   R0498  1          INDEX$_LASTCONT  = DSRINDEX$_LASTCONT  %,
;   R0499  1          INDEX$_NOBEGIN   = DSRINDEX$_NOBEGIN   %,
;   R0500  1          INDEX$_NOEND     = DSRINDEX$_NOEND     %,
;   R0501  1          INDEX$_NOINDEX   = DSRINDEX$_NOINDEX   %,
;   R0502  1          INDEX$_NOLIST    = DSRINDEX$_NOLIST    %,
;   R0503  1          INDEX$_OVERSTRK  = DSRINDEX$_OVERSTRK  %,
;   R0504  1          INDEX$_SKIPPED   = DSRINDEX$_SKIPPED   %,
;   R0505  1          INDEX$_SYNTAX    = DSRINDEX$_SYNTAX    %,
;   R0506  1          INDEX$_TEXFILE   = DSRINDEX$_TEXFILE   %,
;   R0507  1          INDEX$_TOODEEP   = DSRINDEX$_TOODEEP   %,
;   R0508  1          INDEX$_TOOFEW    = DSRINDEX$_TOOFEW    %,
;   R0509  1          INDEX$_TRUNCATED = DSRINDEX$_TRUNCATED %,
;   R0510  1          INDEX$_COMPLETE  = DSRINDEX$_COMPLETE  %,
;   R0511  1          INDEX$_CREATED   = DSRINDEX$_CREATED   %,
;   R0512  1          INDEX$_IDENT     = DSRINDEX$_IDENT     %,
;   R0513  1          INDEX$_PROCFILE  = DSRINDEX$_PROCFILE  %,
;   R0514  1          INDEX$_TEXT      = DSRINDEX$_TEXT      %,
;   R0515  1          INDEX$_TEXTD     = DSRINDEX$_TEXTD     %,
;   R0516  1          INDEX$_TMS11     = DSRINDEX$_TMS11     %;
;   R0517  1
;   R0518  1     %FI
;   R0519  1
;   R0520  1     EXTERNAL LITERAL
;   R0521  1          INDEX$_BADLOGIC,     ! <internal logic error detected>
;   R0522  1          INDEX$_BADVALUE,     ! <'!AS' is an invalid keyword value>
;   R0523  1          INDEX$_INSVIRMEM,    ! <insufficient virtual memory>
;   R0524  1          INDEX$_LINELENG,     ! <maximum line length is 120>
;   R0525  1          INDEX$_NOREF,        ! <page reference not found>
;   R0526  1          INDEX$_OPENIN,       ! <error opening '!AS' for input>
;   R0527  1          INDEX$_OPENOUT,      ! <error opening '!AS' for output>
;   R0528  1          INDEX$_TOOMANY,      ! <too many values supplied>
;   R0529  1          INDEX$_VALERR,       ! <specified value is out of legal range>
;   R0530  1          INDEX$_CANTBAL,      ! <can't balance last page>
```

```
;  R0531  1          INDEX$_CLOSEQUOT,   ! <missing close quote>
;  R0532  1          INDEX$_CONFQUAL,    ! <conflicting qualifiers>
;  R0533  1          INDEX$_CTRLCHAR,    ! <the following line contains control characters - ignored>
;  R0534  1          INDEX$_DOESNTFIT,   ! <'!AD' will not fit at the current indentation level>
;  R0535  1          INDEX$_DUPBEGIN,    ! <duplicate .XPLUS (BEGIN) - inserted as .XPLUS ()>
;  R0536  1          INDEX$_EMPTYIN,     ! <empty input file '!AS'>
;  R0537  1          INDEX$_IGNORED,     ! <'!AS' ignored>
;  R0538  1          INDEX$_INVINPUT,    ! <invalid input file format in file '!AS'>
;  R0539  1          INDEX$_INVRECORD,   ! <invalid record type in file '!AS'>
;  R0540  1          INDEX$_LASTCONT,    ! <can't generate continuation heading on last page>
;  R0541  1          INDEX$_NOBEGIN,     ! <.XPLUS (END) with no .XPLUS (BEGIN) - inserted as .XPLUS ()>
;  R0542  1          INDEX$_NOEND,       ! <.XPLUS (BEGIN) has no corresponding .XPLUS (END)>
;  R0543  1          INDEX$_NOINDEX,     ! <no index information in file '!AS'>
;  R0544  1          INDEX$_NOLIST,      ! <parameter list not allowed>
;  R0545  1          INDEX$_OVERSTRK,    ! <the following line contains an overstrike sequence>
;  R0546  1          INDEX$_SKIPPED,     ! <!UL reference!%S inside page range - ignored>
;  R0547  1          INDEX$_SYNTAX,      ! <error parsing '!AS'>
;  R0548  1          INDEX$_TEXFILE,     ! <error processing line !UL of TEX character file '!AS'>
;  R0549  1          INDEX$_TOODEEP,     ! <maximum subindex depth exceeded>
;  R0550  1          INDEX$_TOOFEW,      ! <not enough values supplied>
;  R0551  1          INDEX$_TRUNCATED,   ! <string too long - truncated>
;  R0552  1          INDEX$_COMPLETE,    ! <processing complete '!AS'>
;  R0553  1          INDEX$_CREATED,     ! <'!AS' created>
;  R0554  1          INDEX$_IDENT,       ! <INDEX version !AD>
;  R0555  1          INDEX$_PROCFILE,    ! <processing file '!AS'>
;  R0556  1          INDEX$_TEXT,        ! <!AS>
;  R0557  1          INDEX$_TEXTD,       ! <entry text: '!AD'>
;  R0558  1          INDEX$_TMS11;       ! <output file full - continuing with file '!AS'>
;  R0559  1
```

```
;  140          0560  1
;  141          0561  1 SWITCHES LIST (NOREQUIRE);
;  142          0562  1
;  143          0563  1 !
;  144          0564  1 ! TABLE OF CONTENTS:
;  145          0565  1 !
;  146          0566  1
;  147          0567  1 FORWARD ROUTINE
;  148          0568  1     NDXCLI,                         ! Command line interface
;  149          0569  1     CONDITION_HANDLER,              ! Main program condition handler - sets return status
;  150          0570  1     CALL_TPARSE,                    ! Invoke TPARSE to parse qualifier values
;  151          0571  1     ENTER_PAGE,                     ! Action routine - enter page number type
;  152          0572  1     OPEN_ERROR;                     ! Report file open errors
;  153          0573  1
;  154        L 0574  1 %IF DSRPLUS
;  155        U 0575  1 %THEN
;  156        U 0576  1
;  157        U 0577  1 FORWARD ROUTINE
;  158        U 0578  1     ENTER_MERGE,                    ! Action routine - enter page range merge
;  159        U 0579  1     ENTER_LAYOUT,                   ! Action routine - enter /LAYOUT value
;  160        U 0580  1     ENTER_FORMAT,                   ! Action routine - enter /FORMAT value
;  161        U 0581  1     ENTER_SORT,                     ! Action routine - enter sort type
;  162        U 0582  1     ENTER_ALPHA,                    ! Action routine - enter nonalpha sort
;  163        U 0583  1     OPTIONS_FILE : NOVALUE,         ! Process options file
;  164        U 0584  1     PARSE_BOOK   : NOVALUE,         ! Process /BOOK_IDENTIFIER qualifier
;  165        U 0585  1     PROCESS_TEX_FILE : NOVALUE,     ! Process TEX character size file
;  166        U 0586  1     STORE_TEX,                      ! Action routine - store TEX character size
;  167        U 0587  1     READ_TEX;                       ! Action routine - read a line from TEX char file
;  168        U 0588  1
;  169          0589  1 %FI
;  170          0590  1
;  171          0591  1 !
;  172          0592  1 ! EQUATED SYMBOLS:
;  173          0593  1 !
;  174          0594  1
;  175          0595  1 LITERAL
;  176          0596  1     TRUE  = 1,
;  177          0597  1     FALSE = 0;
;  178          0598  1
;  179          0599  1 !
;  180          0600  1 ! OWN STORAGE:
;  181          0601  1 !
;  182          0602  1
;  183          0603  1 OWN
;  184          0604  1     VALUE_STR    : $STR_DESCRIPTOR (CLASS = DYNAMIC, STRING = (0, 0)),
;  185          0605  1     OPTIONS_STR : $STR_DESCRIPTOR (CLASS = DYNAMIC, STRING = (0, 0)),
;  186          0606  1     QUALIFIER_VALUE,
;  187          0607  1     TERMINATION_STATUS : INITIAL (STS$K_SUCCESS);
;  188          0608  1
;  189        L 0609  1 %IF DSRPLUS
;  190        U 0610  1 %THEN
;  191        U 0611  1
;  192        U 0612  1 OWN
;  193        U 0613  1     TEX_FILE_NAME : $STR_DESCRIPTOR (CLASS = DYNAMIC, STRING = (0, 0)),
;  194        U 0614  1     TEX_CHAR_SIZES : VECTOR [256],          ! Where character sizes are stroed
;  195        U 0615  1     TEX_CHAR_INDEX,                         ! Index into TEX_CHAR_SIZES
;  196        U 0616  1     TEX_FILE_LINE_NO,                       ! Line number of file
```

```
;  197          U 0617  1     TEX_LINE : $STR_DESCRIPTOR (),                  ! Descriptor of input line
;  198          U 0618  1     TEX_IN_BUF : BLOCK [512, BYTE],                 ! Input buffer
;  199          U 0619  1     TEX_ES : BLOCK [NAM$C_MAXRSS, BYTE],            ! Expanded filename string
;  200          U 0620  1     TEX_RS : BLOCK [NAM$C_MAXRSS, BYTE],            ! Resultant filename string
;  201          U 0621  1     TEX_NAM : $NAM (ESA = TEX_ES, ESS = NAM$C_MAXRSS, RSA = TEX_RS, RSS = NAM$C_MAXRSS),
;  202          U 0622  1     TEX_FAB : $FAB (NAM = TEX_NAM, DNM = '.FSZ'),
;  203          U 0623  1     TEX_RAB : $RAB (FAB = TEX_FAB, UBF = TEX_IN_BUF, USZ = 512);
;  204          U 0624  1
;  205            0625  1 %FI
;  206            0626  1
;  207            0627  1 !
;  208            0628  1 ! EXTERNAL REFERENCES:
;  209            0629  1 !
;  210            0630  1
;  211            0631  1 EXTERNAL LITERAL
;  212            0632  1     TAB : UNSIGNED (8),             ! TAB character
;  213            0633  1     TMSCOL,                         ! Default TMS column width
;  214            0634  1     MAXLIN;                         ! Maximum number of lines per page
;  215            0635  1
;  216            0636  1 EXTERNAL LITERAL                    ! Values returned from CLI interface
;  217            0637  1     CLI$_CONCAT,                    ! Value concatenated to next
;  218            0638  1     CLI$_PRESENT,                   ! Value explicitly given
;  219            0639  1     CLI$_NEGATED,                   ! Value explicitly negated (/NO)
;  220            0640  1     CLI$_DEFAULTED,                 ! Value defaulted present
;  221            0641  1     CLI$_ABSENT;                    ! Value not present
;  222            0642  1
;  223            0643  1 EXTERNAL
;  224            0644  1     CMDBLK : $NDXCMD,               ! Command line information block
;  225            0645  1     CHRSIZ : REF VECTOR,            ! TMS character size vector pointer
;  226            0646  1     CHRSZA : VECTOR,                ! Character size vector for /TMS11 = A
;  227            0647  1     CHRSZE : VECTOR,                ! Character size vector for /TMS11 = E
;  228            0648  1     NDXVRL,                         ! Length of version number string
;  229            0649  1     NDXVRP;                         ! CH$PTR to version number string
;  230            0650  1
;  231            0651  1 EXTERNAL ROUTINE
;  232            0652  1     NDXINI          : NOVALUE,                          ! Once only initialization
;  233            0653  1     NDXINP          : NOVALUE,                          ! Process input file
;  234            0654  1     MAKNDX          : NOVALUE,                          ! Generate index
;  235            0655  1     CLI$PRESENT     : ADDRESSING_MODE (GENERAL),        ! Check for qualifier
;  236            0656  1     CLI$GET_VALUE   : ADDRESSING_MODE (GENERAL),        ! Get value of qualifier
;  237            0657  1     LIB$TPARSE      : ADDRESSING_MODE (GENERAL);        ! Table driven parser
;  238            0658  1
;  239          L 0659  1 %IF DSRPLUS
;  240          U 0660  1 %THEN
;  241          U 0661  1
;  242          U 0662  1 EXTERNAL
;  243          U 0663  1     NDXOPTION;                    ! Options file parse tables address
;  244          U 0664  1
;  245          U 0665  1 EXTERNAL ROUTINE
;  246          U 0666  1     CLI$DCL_PARSE   : ADDRESSING_MODE (GENERAL);        ! Initiate new parse
;  247          U 0667  1
;  248            0668  1 %FI
;  249            0669  1
;  250            0670  1 !+
;  251            0671  1 !   TPARSE state tables
;  252            0672  1 !-
;  253            0673  1
```

```
;  254          0674  1 !
;  255          0675  1 ! Tables to parse an arbitrary number
;  256          0676  1 !
;  257          0677  1 $INIT_STATE (NUMBER_STATE, NUMBER_KEY);
;  258        P 0678  1 $STATE (
;  259        P 0679  1     (TPA$_DECIMAL,          ,              , QUALIFIER_VALUE),
;  260        P 0680  1     (TPA$_EOS,      TPA$_EXIT)
;  261          0681  1     );
;  262        P 0682  1 $STATE (,
;  263        P 0683  1     (TPA$_EOS,      TPA$_EXIT)
;  264          0684  1     );
;  265          0685  1
;  266          0686  1 !
;  267          0687  1 ! Tables to parse /PAGE_NUMBERS values
;  268          0688  1 !
;  269          0689  1 $INIT_STATE (PAGE_STATE, PAGE_KEY);
;  270        P 0690  1 $STATE (,
;  271        P 0691  1
;  272        L 0692  1 %IF DSRPLUS
;  273        U 0693  1 %THEN
;  274        U 0694  1
;  275        U 0695  1     ('MERGE',       ,          ENTER_MERGE,  , ,          TRUE),
;  276        U 0696  1     ('NOMERGE',     ,          ENTER_MERGE,  , ,          FALSE),
;  277        U 0697  1     ('STANDARD',    ,          ENTER_PAGE,   , ,          TRUE),
;  278        U 0698  1
;  279        P 0699  1 %FI
;  280        P 0700  1
;  281        P 0701  1     ('RUNNING',     ,          ENTER_PAGE,   , ,          FALSE),
;  282        P 0702  1     ('NORUNNING',   ,          ENTER_PAGE,   , ,          TRUE)
;  283          0703  1     );
;  284        P 0704  1 $STATE (,
;  285        P 0705  1     (TPA$_EOS,      TPA$_EXIT)
;  286          0706  1     );
;  287          0707  1
;  288        L 0708  1 %IF DSRPLUS
;  289        U 0709  1 %THEN
;  290        U 0710  1
;  291        U 0711  1 !
;  292        U 0712  1 ! Tables to parse /FORMAT values
;  293        U 0713  1 !
;  294        U 0714  1 $INIT_STATE (FORMAT_STATE, FORMAT_KEY);
;  295        U 0715  1 $STATE (,
;  296        U 0716  1     ('DSR',         FORMAT_END, ENTER_FORMAT, , ,          DSR),
;  297        U 0717  1     ('TEX',         TEX_STATE),
;  298        U 0718  1     ('TMS',         TMS_STATE)
;  299        U 0719  1     );
;  300        U 0720  1 $STATE (TEX_STATE,
;  301        U 0721  1     ('=',           TPA$_EXIT,  ENTER_FORMAT, , ,          TEX),
;  302        U 0722  1     (':',           TPA$_EXIT,  ENTER_FORMAT, , ,          TEX)
;  303        U 0723  1     );
;  304        U 0724  1 $STATE (TMS_STATE,
;  305        U 0725  1     ('='),
;  306        U 0726  1     (':'),
;  307        U 0727  1     (TPA$_EOS,      TPA$_EXIT,  ENTER_FORMAT, , ,          TMS11_A)
;  308        U 0728  1     );
;  309        U 0729  1 $STATE (,
;  310        U 0730  1     ('A',           FORMAT_END, ENTER_FORMAT, , ,          TMS11_A),
```

```
;  311        U 0731  1      ('E',          FORMAT_END, ENTER_FORMAT, , ,                TMS11_E)
;  312        U 0732  1      );
;  313        U 0733  1  $STATE (FORMAT_END,
;  314        U 0734  1      (TPA$_EOS,     TPA$_EXIT)
;  315        U 0735  1      );
;  316        U 0736  1
;  317        U 0737  1  !
;  318        U 0738  1  ! Tables to parse /LAYOUT values
;  319        U 0739  1  !
;  320        U 0740  1  $INIT_STATE (LAYOUT_STATE, LAYOUT_KEY);
;  321        U 0741  1  $STATE (,
;  322        U 0742  1      ('TWO_COLUMN', LAYOUT_END, ENTER_LAYOUT, , ,                TWO_COLUMN),
;  323        U 0743  1      ('2',          LAYOUT_END, ENTER_LAYOUT, , ,                TWO_COLUMN),
;  324        U 0744  1      ('ONE_COLUMN', LAYOUT_END, ENTER_LAYOUT, , ,                ONE_COLUMN),
;  325        U 0745  1      ('1',          LAYOUT_END, ENTER_LAYOUT, , ,                ONE_COLUMN),
;  326        U 0746  1      ('GALLEY',     LAYOUT_END, ENTER_LAYOUT, , ,                GALLEY),
;  327        U 0747  1      ('SEPARATE',   ,           ENTER_LAYOUT, , ,                SEPARATE)
;  328        U 0748  1      );
;  329        U 0749  1  $STATE (,
;  330        U 0750  1      ('='),
;  331        U 0751  1      (':'),
;  332        U 0752  1      (TPA$_EOS,     TPA$_EXIT)
;  333        U 0753  1      );
;  334        U 0754  1  $STATE (,
;  335        U 0755  1      (TPA$_DECIMAL, LAYOUT_END, ,            , QUALIFIER_VALUE)
;  336        U 0756  1      );
;  337        U 0757  1  $STATE (LAYOUT_END,
;  338        U 0758  1      (TPA$_EOS,     TPA$_EXIT)
;  339        U 0759  1      );
;  340        U 0760  1
;  341        U 0761  1  !
;  342        U 0762  1  ! Tables to parse /SORT values
;  343        U 0763  1  !
;  344        U 0764  1  $INIT_STATE (SORT_STATE, SORT_KEY);
;  345        U 0765  1  $STATE (,
;  346        U 0766  1      ('WORD',       SORT_END,   ENTER_SORT,   : :                TRUE),
;  347        U 0767  1      ('LETTER',     SORT_END,   ENTER_SORT,   : :                FALSE),
;  348        U 0768  1      ('NONALPHA')
;  349        U 0769  1      );
;  350        U 0770  1  $STATE (,
;  351        U 0771  1      ('='),
;  352        U 0772  1      (':')
;  353        U 0773  1      );
;  354        U 0774  1  $STATE (,
;  355        U 0775  1      ('IGNORE',     SORT_END,   ENTER_ALPHA,  : :                IGNORE),
;  356        U 0776  1      ('BEFORE',     SORT_END,   ENTER_ALPHA,  : :                BEFORE),
;  357        U 0777  1      ('AFTER',      SORT_END,   ENTER_ALPHA,  : :                AFTER)
;  358        U 0778  1      );
;  359        U 0779  1  $STATE (SORT_END,
;  360        U 0780  1      (TPA$_EOS,     TPA$_EXIT)
;  361        U 0781  1      );
;  362        U 0782  1
;  363        U 0783  1  !
;  364        U 0784  1  ! Tables to parse TEX character size file
;  365        U 0785  1  !
;  366        U 0786  1  $INIT_STATE (TEX_FILE_STATE, TEX_FILE_KEY);
;  367        U 0787  1  $STATE (TEX_1,
```

```
;  368        U 0788  1      ('!',                TEX_1,      READ_TEX),
;  369        U 0789  1      (TPA$_EOS,           TEX_1,      READ_TEX),
;  370        U 0790  1      (TPA$_DECIMAL,       TEX_2,      STORE_TEX)
;  371        U 0791  1      );
;  372        U 0792  1  $STATE (TEX_2,
;  373        U 0793  1      (' ',                TEX_1),
;  374        U 0794  1      ('!',                TEX_2,      READ_TEX),
;  375        U 0795  1      (TPA$_EOS,           TEX_2,      READ_TEX)
;  376        U 0796  1      );
;  377        U 0797  1
;  378          0798  1  %FI
```

```
;  380          0799  1 %SBTTL 'NDXCLI -- Main program - command line interface'
;  381          0800  1 GLOBAL ROUTINE NDXCLI =
;  382          0801  1 !++
;  383          0802  1 !
;  384          0803  1 ! FUNCTIONAL DESCRIPTION:
;  385          0804  1 !
;  386          0805  1 !       This routine uses the VMS DCL CLE to obtain command
;  387          0806  1 !       line information which is in turn passed to the INDEX
;  388          0807  1 !       application in a transportable manner.
;  389          0808  1 !
;  390          0809  1 ! FORMAL PARAMETERS:
;  391          0810  1 !
;  392          0811  1 !       None
;  393          0812  1 !
;  394          0813  1 ! IMPLICIT INPUTS:
;  395          0814  1 !
;  396          0815  1 !       None
;  397          0816  1 !
;  398          0817  1 ! IMPLICIT OUTPUTS:
;  399          0818  1 !
;  400          0819  1 !       CMDBLK - The command line information block is filled in
;  401          0820  1 !
;  402          0821  1 ! ROUTINE VALUE:
;  403          0822  1 ! COMPLETION CODES:
;  404          0823  1 !
;  405          0824  1 !       TERMINATION_STATUS      - Set by CONDITION_HANDLER ()
;  406          0825  1 !
;  407          0826  1 ! SIDE EFFECTS:
;  408          0827  1 !
;  409          0828  1 !       None
;  410          0829  1 !
;  411          0830  1 !--
;  412          0831  1
;  413          0832  2     BEGIN
;  414          0833  2
;  415          0834  2     ENABLE
;  416          0835  2         CONDITION_HANDLER;
;  417          0836  2
;  418          0837  2     LOCAL
;  419          0838  2         STATUS;
;  420          0839  2
;  421          0840  2     NDXINI ();                  ! Do once-only initialization
;  422          0841  2
;  423          0842  2     !
;  424          0843  2     ! Get copy of whole command line
;  425          0844  2     !
;  426          0845  2     CLI$GET_VALUE (%ASCID'$LINE', CMDBLK [NDX$T_COMMAND_LINE]);
;  427          0846  2
;  428          0847  2     !
;  429          0848  2     ! /[NO]MASTER
;  430          0849  2     !
;  431          0850  2     !                       * W A R N I N G *
;  432          0851  2     !
;  433          0852  2     !               This must be parsed before other qualifiers.
;  434          0853  2     !               Other qualifiers depend on the value of this qualifier.
;  435          0854  2     !
;  436          0855  2     !                       * W A R N I N G *
```

```
;  437          0856  2      !
;  438          0857  2      CMDBLK [NDX$V_MASTER] = FALSE;
;  439          0858  2
;  440        L 0859  2  %IF DSRPLUS
;  441        U 0860  2  %THEN
;  442        U 0861  2
;  443        U 0862  2      IF  CLI$PRESENT (%ASCID'MASTER')
;  444        U 0863  2      THEN
;  445        U 0864  2          CMDBLK [NDX$V_MASTER] = TRUE;
;  446          0865  2  %FI
;  447          0866  2
;  448          0867  2      !
;  449          0868  2      ! /FORMAT = { DSR ! TEX : filename ! TMS11 [ = { A ! E }] }
;  450          0869  2      !
;  451          0870  2      !                       * W A R N I N G *
;  452          0871  2      !
;  453          0872  2      !               This must be parsed before other qualifiers.
;  454          0873  2      !               Other qualifiers depend on the value of this qualifier.
;  455          0874  2      !
;  456          0875  2      !                       * W A R N I N G *
;  457          0876  2      !
;  458          0877  2      CMDBLK [NDX$H_FORMAT] = DSR;        ! Assume output for RUNOFF
;  459          0878  2      CHRSIZ                = CHRSZA;     ! Assume TMS11 type 'A' characters
;  460          0879  2
;  461        L 0880  2  %IF DSRPLUS
;  462        U 0881  2  %THEN
;  463        U 0882  2
;  464        U 0883  2      IF  CLI$PRESENT (%ASCID'FORMAT')
;  465        U 0884  2      THEN
;  466        U 0885  2          BEGIN
;  467        U 0886  2          CLI$GET_VALUE (%ASCID'FORMAT', VALUE_STR);
;  468        U 0887  2
;  469        U 0888  2          IF  NOT CALL_TPARSE (VALUE_STR, FORMAT_STATE, FORMAT_KEY)
;  470        U 0889  2          THEN
;  471        U 0890  2              SIGNAL_STOP (INDEX$_BADVALUE, 1, VALUE_STR);
;  472        U 0891  2
;  473        U 0892  2          IF .CMDBLK [NDX$H_FORMAT] EQL TEX THEN PROCESS_TEX_FILE ();
;  474        U 0893  2          END;
;  475        U 0894  2
;  476          0895  2  %FI
;  477          0896  2
;  478          0897  2      !
;  479          0898  2      ! /COLUMN_WIDTH = n
;  480          0899  2      !
;  481          0900  2      !                       * W A R N I N G *
;  482          0901  2      !
;  483          0902  2      !               This must be parsed after /FORMAT and before any other
;  484          0903  2      !               qualifier. It depends on the value of /FORMAT and other
;  485          0904  2      !               qualifiers depend on the value of this qualifier.
;  486          0905  2      !
;  487          0906  2      !                       * W A R N I N G *
;  488          0907  2      !
;  489          0908  2      CMDBLK [NDX$G_COLUMN_WID] = 34;             ! Default column width is 34
;  490          0909  2
;  491        L 0910  2  %IF DSRPLUS
;  492        U 0911  2  %THEN
;  493        U 0912  2
```

```
;  494        U 0913  2      IF  CLI$PRESENT (%ASCID'COLUMN_WIDTH')
;  495        U 0914  2      THEN
;  496        U 0915  2          BEGIN
;  497        U 0916  2          QUALIFIER_VALUE = 0;
;  498        U 0917  2          CLI$GET_VALUE (%ASCID'COLUMN_WIDTH', VALUE_STR);
;  499        U 0918  2
;  500        U 0919  2          IF  NOT CALL_TPARSE (VALUE_STR, NUMBER_STATE, NUMBER_KEY)
;  501        U 0920  2          THEN
;  502        U 0921  2              SIGNAL_STOP (INDEX$_BADVALUE, 1, VALUE_STR);
;  503        U 0922  2
;  504        U 0923  2          CMDBLK [NDX$G_COLUMN_WID] = .QUALIFIER_VALUE;
;  505        U 0924  2
;  506        U 0925  2          IF  .CMDBLK [NDX$G_COLUMN_WID] LSS 5
;  507        U 0926  2          THEN
;  508        U 0927  2              SIGNAL_STOP (INDEX$_BADVALUE, 1, VALUE_STR, INDEX$_VALERR);
;  509        U 0928  2
;  510        U 0929  2          END
;  511        U 0930  2      ELSE
;  512        U 0931  2          BEGIN
;  513        U 0932  2
;  514        U 0933  2          IF .CMDBLK [NDX$H_FORMAT] NEQ DSR
;  515        U 0934  2          THEN
;  516        U 0935  2              !
;  517        U 0936  2              ! Typeset column width default is defined by the literal TMSCOL
;  518        U 0937  2              !
;  519        U 0938  2              CMDBLK [NDX$G_COLUMN_WID] = TMSCOL;
;  520        U 0939  2
;  521        U 0940  2          END;
;  522        U 0941  2
;  523          0942  2  %FI
;  524          0943  2
;  525          0944  2      !
;  526          0945  2      ! /LAYOUT = ( TWO_COLUMN | ONE_COLUMN | GALLEY | SEPARATE [= n] )
;  527          0946  2      !
;  528          0947  2      !                       * W A R N I N G *
;  529          0948  2      !
;  530          0949  2      !               This must be parsed after /COLUMN_WIDTH and before any other
;  531          0950  2      !               qualifier. It depends on the value of /COLUMN_WIDTH and other
;  532          0951  2      !               qualifiers depend on the value of this qualifier.
;  533          0952  2      !
;  534          0953  2      !                       * W A R N I N G *
;  535          0954  2      !
;  536          0955  2      CMDBLK [NDX$H_LAYOUT]          = TWO_COLUMN; ! Default index layout
;  537          0956  2      CMDBLK [NDX$G_SEPARATE_WIDTH] = .CMDBLK [NDX$G_COLUMN_WID];
;  538          0957  2
;  539        L 0958  2  %IF DSRPLUS
;  540        U 0959  2  %THEN
;  541        U 0960  2
;  542        U 0961  2      IF  CLI$PRESENT (%ASCID'LAYOUT')
;  543        U 0962  2      THEN
;  544        U 0963  2          BEGIN
;  545        U 0964  2          QUALIFIER_VALUE = -1;
;  546        U 0965  2          CLI$GET_VALUE (%ASCID'LAYOUT', VALUE_STR);
;  547        U 0966  2
;  548        U 0967  2          IF  NOT CALL_TPARSE (VALUE_STR, LAYOUT_STATE, LAYOUT_KEY)
;  549        U 0968  2          THEN
;  550        U 0969  2              SIGNAL_STOP (INDEX$_BADVALUE, 1, VALUE_STR);
```

```
;   551        U 0970  2
;   552        U 0971  2          IF  .QUALIFIER_VALUE NEQ -1
;   553        U 0972  2          THEN
;   554        U 0973  2              BEGIN
;   555        U 0974  2              !
;   556        U 0975  2              ! Doing SEPARATE index and user specified reference column width.
;   557        U 0976  2              ! Validate against minimum column width.
;   558        U 0977  2              !
;   559        U 0978  2
;   560        U 0979  2              IF  .QUALIFIER_VALUE LSS 5
;   561        U 0980  2              THEN
;   562        U 0981  2                  SIGNAL_STOP (INDEX$_BADVALUE, 1, VALUE_STR, INDEX$_VALERR);
;   563        U 0982  2
;   564        U 0983  2              CMDBLK [NDX$G_SEPARATE_WIDTH] = .QUALIFIER_VALUE;
;   565        U 0984  2              END;
;   566        U 0985  2
;   567        U 0986  2          END;
;   568        U 0987  2
;   569        U 0988  2      IF  (.CMDBLK [NDX$H_FORMAT] EQL TEX)
;   570        U 0989  2      AND (.CMDBLK [NDX$H_LAYOUT] NEQ TWO_COLUMN)
;   571        U 0990  2      THEN
;   572        U 0991  2          BEGIN
;   573        U 0992  2
;   574        U 0993  2          LOCAL
;   575        U 0994  2              FORMAT_PTR;
;   576        U 0995  2
;   577        U 0996  2          FORMAT_PTR = (
;   578        U 0997  2              SELECTONE .CMDBLK [NDX$H_LAYOUT] OF
;   579        U 0998  2                  SET
;   580        U 0999  2
;   581        U 1000  2                  [ONE_COLUMN]:   %ASCID 'ONE_COLUMN';
;   582        U 1001  2                  [GALLEY]:       %ASCID 'GALLEY';
;   583        U 1002  2                  [SEPARATE]:     %ASCID 'SEPARATE';
;   584        U 1003  2
;   585        U 1004  2                  TES
;   586        U 1005  2              );
;   587        U 1006  2
;   588        U 1007  2          SIGNAL_STOP (INDEX$_BADVALUE, 1, .FORMAT_PTR, INDEX$_CONFQUAL);
;   589        U 1008  2          END;
;   590        U 1009  2
;   591           1010  2 %FI
;   592           1011  2
;   593           1012  2      !
;   594           1013  2      ! /[NO]TELLTALE_HEADINGS
;   595           1014  2      !
;   596           1015  2      !                       * W A R N I N G *
;   597           1016  2      !
;   598           1017  2      !               This must be parsed after /LAYOUT and before /LINES_PER_PAGE
;   599           1018  2      !               It depends on the value of /LAYOUT and /LINES_PER_PAGE
;   600           1019  2      !               depends on the value of this qualifier.
;   601           1020  2      !
;   602           1021  2      !                       * W A R N I N G *
;   603           1022  2      !
;   604           1023  2      CMDBLK [NDX$V_TELLTALE] = FALSE;
;   605           1024  2
;   606        L  1025  2 %IF DSRPLUS
;   607        U  1026  2 %THEN
```

```
;  608     U 1027  2
;  609     U 1028  2       IF  CLISPRESENT (%ASCID'TELLTALE_HEADINGS')
;  610     U 1029  2       THEN
;  611     U 1030  2           BEGIN
;  612     U 1031  2
;  613     U 1032  2           IF  .CMDBLK [NDX$H_LAYOUT] EQL GALLEY
;  614     U 1033  2           THEN
;  615     U 1034  2               !
;  616     U 1035  2               ! Doing TMS11 galley output.
;  617     U 1036  2               ! Telltale headings are not allowed
;  618     U 1037  2               !
;  619     U 1038  2               SIGNAL (INDEX$_IGNORED, 1, %ASCID'TELLTALE_HEADINGS', INDEX$_CONFQUAL)
;  620     U 1039  2           ELSE
;  621     U 1040  2               CMDBLK [NDX$V_TELLTALE] = TRUE;
;  622     U 1041  2
;  623     U 1042  2           END;
;  624     U 1043  2
;  625       1044  2   %FI
;  626       1045  2
;  627       1046  2       !
;  628       1047  2       ! /LINES_PER_PAGE = n
;  629       1048  2       !
;  630       1049  2       !                       * W A R N I N G *
;  631       1050  2       !
;  632       1051  2       !               This must be parsed after /FORMAT, /LAYOUT and
;  633       1052  2       !               /TELLTALE_HEADINGS. It depends on the value of these qualifiers
;  634       1053  2       !
;  635       1054  2       !                       * W A R N I N G *
;  636       1055  2       !
;  637       1056  2       IF .CMDBLK [NDX$H_FORMAT] EQL DSR
;  638       1057  2       THEN
;  639       1058  3           BEGIN                                       ! Formatting for RUNOFF
;  640       1059  3
;  641       1060  3           IF .CMDBLK [NDX$V_TELLTALE]
;  642       1061  3           THEN
;  643       1062  3               CMDBLK [NDX$G_LINES_PAGE] = 52          ! 52 lines with /TELLTALE
;  644       1063  3           ELSE
;  645       1064  3               CMDBLK [NDX$G_LINES_PAGE] = 55;         ! 55 lines per page otherwise
;  646       1065  3
;  647       1066  3           END
;  648       1067  2       ELSE
;  649       1068  2           CMDBLK [NDX$G_LINES_PAGE] = 54;             ! 54 lines per page for Typeset
;  650       1069  2
;  651       1070  2       IF  CLI$PRESENT (%ASCID'LINES_PER_PAGE')
;  652       1071  2       THEN
;  653       1072  3           BEGIN
;  654       1073  3           !
;  655       1074  3           ! User specified a value
;  656       1075  3           !
;  657       1076  3
;  658       1077  3           IF  .CMDBLK [NDX$H_LAYOUT] EQL GALLEY
;  659       1078  3           THEN
;  660       1079  3               !
;  661       1080  3               ! Galley output - ignore lines-per-page
;  662       1081  3               !
;  663       1082  3               SIGNAL (INDEX$_IGNORED, 1, %ASCID'LINES_PER_PAGE', INDEX$_CONFQUAL)
;  664       1083  3           ELSE
```

```
;  665          1084  4              BEGIN
;  666          1085  4              QUALIFIER_VALUE = 0;
;  667          1086  4              CLI$GET_VALUE (%ASCID'LINES_PER_PAGE', VALUE_STR);
;  668          1087  4
;  669          1088  4              IF  NOT CALL_TPARSE (VALUE_STR, NUMBER_STATE, NUMBER_KEY)
;  670          1089  4              THEN
;  671          1090  4                  SIGNAL_STOP (INDEX$_BADVALUE, 1, VALUE_STR);
;  672          1091  4
;  673          1092  4              CMDBLK [NDX$G_LINES_PAGE] = .QUALIFIER_VALUE;
;  674          1093  4
;  675          1094  5              IF  (
;  676          1095  6                  (.CMDBLK [NDX$G_LINES_PAGE] LSS 15)
;  677          1096  6                  AND (.CMDBLK [NDX$H_FORMAT] EQL DSR)
;  678          1097  5                  )
;  679          1098  5              OR  (
;  680          1099  6                  (.CMDBLK [NDX$G_LINES_PAGE] LSS 25)
;  681          1100  6                  AND (.CMDBLK [NDX$H_FORMAT] NEQ DSR)
;  682          1101  5                  )
;  683          1102  5              OR  (.CMDBLK [NDX$G_LINES_PAGE] GTR MAXLIN)
;  684          1103  4              THEN
;  685          1104  4                  SIGNAL_STOP (INDEX$_BADVALUE, 1, VALUE_STR, INDEX$_VALERR);
;  686          1105  4
;  687          1106  3              END;
;  688          1107  3
;  689          1108  2          END;
;  690          1109  2
;  691          1110  2      !
;  692          1111  2      ! /GUTTER_WIDTH = n
;  693          1112  2      !
;  694          1113  2      !                       * W A R N I N G *
;  695          1114  2      !
;  696          1115  2      !                 This qualifier depends on the value of /LAYOUT
;  697          1116  2      !
;  698          1117  2      !                       * W A R N I N G *
;  699          1118  2      !
;  700          1119  2          CMDBLK [NDX$G_GUTTER_WID] = 2;
;  701          1120  2
;  702        L 1121  2      %IF DSRPLUS
;  703        U 1122  2      %THEN
;  704        U 1123  2
;  705        U 1124  2          IF  (.CMDBLK [NDX$H_LAYOUT] EQL ONE_COLUMN) OR
;  706        U 1125  2              (.CMDBLK [NDX$H_LAYOUT] EQL GALLEY)
;  707        U 1126  2          THEN
;  708        U 1127  2              BEGIN
;  709        U 1128  2              !
;  710        U 1129  2              ! ONE_COLUMN output which is not a separate master index
;  711        U 1130  2              ! or GALLEY output
;  712        U 1131  2              !
;  713        U 1132  2              CMDBLK [NDX$G_GUTTER_WID] = 0;          ! Gutter width is meaningless
;  714        U 1133  2
;  715        U 1134  2              IF  CLI$PRESENT (%ASCID'GUTTER_WIDTH')
;  716        U 1135  2              THEN
;  717        U 1136  2                  SIGNAL (INDEX$_IGNORED, 1, %ASCID'GUTTER_WIDTH', INDEX$_CONFQUAL);
;  718        U 1137  2
;  719        U 1138  2              END
;  720        U 1139  2          ELSE
;  721        U 1140  2              BEGIN
```

```
;  722          U 1141  2          !
;  723          U 1142  2          ! For all other page layouts
;  724          U 1143  2          !
;  725          U 1144  2          QUALIFIER_VALUE = 2;                        ! Default value
;  726          U 1145  2
;  727          U 1146  2          IF  CLI$PRESENT (%ASCID'GUTTER_WIDTH')
;  728          U 1147  2          THEN
;  729          U 1148  2              BEGIN
;  730          U 1149  2              CLI$GET_VALUE (%ASCID'GUTTER_WIDTH', VALUE_STR);
;  731          U 1150  2
;  732          U 1151  2              IF  NOT CALL_TPARSE (VALUE_STR, NUMBER_STATE, NUMBER_KEY)
;  733          U 1152  2              THEN
;  734          U 1153  2                  SIGNAL_STOP (INDEX$_BADVALUE, 1, VALUE_STR);
;  735          U 1154  2
;  736          U 1155  2              END;
;  737          U 1156  2
;  738          U 1157  2          CMDBLK [NDX$G_GUTTER_WID] = .QUALIFIER_VALUE;
;  739          U 1158  2          END;
;  740          U 1159  2
;  741            1160  2  %FI
;  742            1161  2
;  743            1162  2      !
;  744            1163  2      ! Validate the combinations of column width, gutter width, and
;  745            1164  2      ! right column width for master indexes.
;  746            1165  2      !
;  747            1166  2      !                       * W A R N I N G *
;  748            1167  2      !
;  749            1168  2      !               This code depends on the value of /LAYOUT, /COLUMN_WIDTH
;  750            1169  2      !               and /GUTTER_WIDTH
;  751            1170  2      !
;  752            1171  2      !                       * W A R N I N G *
;  753            1172  2      !
;  754            1173  2      SELECTONE .CMDBLK [NDX$H_LAYOUT] OF
;  755            1174  2      SET
;  756            1175  2
;  757            1176  2      [TWO_COLUMN]:
;  758            1177  2
;  759            1178  2          IF  (2 * .CMDBLK [NDX$G_COLUMN_WID]) +
;  760            1179  2              .CMDBLK [NDX$G_GUTTER_WID] GTR 120
;  761            1180  2          THEN
;  762            1181  2              SIGNAL_STOP (INDEX$_LINELENG);
;  763            1182  2
;  764            1183  2      [SEPARATE]:
;  765            1184  2
;  766            1185  2          IF  .CMDBLK [NDX$G_COLUMN_WID] +
;  767            1186  2              .CMDBLK [NDX$G_GUTTER_WID] +
;  768            1187  2              .CMDBLK [NDX$G_SEPARATE_WIDTH] GTR 120
;  769            1188  2          THEN
;  770            1189  2              SIGNAL_STOP (INDEX$_LINELENG);
;  771            1190  2
;  772            1191  2      [OTHERWISE]:
;  773            1192  2
;  774            1193  2          IF  .CMDBLK [NDX$G_COLUMN_WID] GTR 120
;  775            1194  2          THEN
;  776            1195  2              SIGNAL_STOP (INDEX$_LINELENG);
;  777            1196  2
;  778            1197  2      TES;
```

```
;   779          1198  2
;   780          1199  2
;   781          1200  2        ! /[NO]CONTINUATION_HEADINGS
;   782          1201  2        !
;   783          1202  2        !                       * W A R N I N G *
;   784          1203  2        !
;   785          1204  2        !             This qualifier depends on the value of /LAYOUT
;   786          1205  2        !
;   787          1206  2        !                       * W A R N I N G *
;   788          1207  2        !
;   789          1208  2        CMDBLK [NDX$V_CONTINUATION] = FALSE;
;   790          1209  2
;   791        L 1210  2    %IF DSRPLUS
;   792        U 1211  2    %THEN
;   793        U 1212  2
;   794        U 1213  2        IF  CLI$PRESENT (%ASCID'CONTINUATION_HEADINGS')
;   795        U 1214  2        THEN
;   796        U 1215  2            BEGIN
;   797        U 1216  2
;   798        U 1217  2            IF  .CMDBLK [NDX$H_LAYOUT] EQL GALLEY
;   799        U 1218  2            THEN
;   800        U 1219  2                !
;   801        U 1220  2                ! Doing TMS11 galley output.
;   802        U 1221  2                ! Continuation headings are not allowed
;   803        U 1222  2                !
;   804        U 1223  2                SIGNAL (INDEX$_IGNORED, 1, %ASCID'CONTINUATION_HEADINGS', INDEX$_CONFQUAL)
;   805        U 1224  2            ELSE
;   806        U 1225  2                CMDBLK [NDX$V_CONTINUATION] = TRUE;
;   807        U 1226  2
;   808        U 1227  2            END;
;   809        U 1228  2
;   810          1229  2    %FI
;   811          1230  2
;   812          1231  2        !
;   813          1232  2        ! /NORESERVE
;   814          1233  2        ! /RESERVE = n
;   815          1234  2        !
;   816          1235  2        !                       * W A R N I N G *
;   817          1236  2        !
;   818          1237  2        !             This qualifier depends on the value of /LINES_PER_PAGE
;   819          1238  2        !
;   820          1239  2        !                       * W A R N I N G *
;   821          1240  2        !
;   822          1241  2        CMDBLK [NDX$G_RESERVE_LINES] = 0;
;   823          1242  2
;   824          1243  2        IF  CLI$PRESENT (%ASCID'RESERVE')
;   825          1244  2        THEN
;   826          1245  3            BEGIN
;   827          1246  3            QUALIFIER_VALUE = 0;
;   828          1247  3            CLI$GET_VALUE (%ASCID'RESERVE', VALUE_STR);
;   829          1248  3
;   830          1249  3            IF  NOT CALL_TPARSE (VALUE_STR, NUMBER_STATE, NUMBER_KEY)
;   831          1250  3            THEN
;   832          1251  3                SIGNAL_STOP (INDEX$_BADVALUE, 1, VALUE_STR);
;   833          1252  3
;   834          1253  3            IF  .QUALIFIER_VALUE GTR .CMDBLK [NDX$G_LINES_PAGE]
;   835          1254  3            THEN
```

```
;  836          1255  3              SIGNAL_STOP (INDEX$_BADVALUE, 1, VALUE_STR, INDEX$_VALERR);
;  837          1256  3
;  838          1257  3          CMDBLK [NDX$G_RESERVE_LINES] = .QUALIFIER_VALUE;
;  839          1258  2          END;
;  840          1259  2
;  841          1260  2      !
;  842          1261  2      ! /LEVEL = n
;  843          1262  2      !
;  844          1263  2      !                       * W A R N I N G *
;  845          1264  2      !
;  846          1265  2      !             This qualifier depends on the value of /MASTER
;  847          1266  2      !
;  848          1267  2      !                       * W A R N I N G *
;  849          1268  2      !
;  850          1269  2      CMDBLK [NDX$H_LEVEL] = 99;                  ! All levels
;  851          1270  2
;  852        L 1271  2  %IF DSRPLUS
;  853        U 1272  2  %THEN
;  854        U 1273  2
;  855        U 1274  2      IF  CLI$PRESENT (%ASCID'LEVEL')
;  856        U 1275  2      THEN
;  857        U 1276  2          BEGIN
;  858        U 1277  2          CLI$GET_VALUE (%ASCID'LEVEL', VALUE_STR);
;  859        U 1278  2
;  860        U 1279  2          IF  NOT CALL_TPARSE (VALUE_STR, NUMBER_STATE, NUMBER_KEY)
;  861        U 1280  2          THEN
;  862        U 1281  2              SIGNAL_STOP (INDEX$_BADVALUE, 1, VALUE_STR);
;  863        U 1282  2
;  864        U 1283  2          IF  .QUALIFIER_VALUE LEQ 0
;  865        U 1284  2          THEN
;  866        U 1285  2              SIGNAL_STOP (INDEX$_BADVALUE, 1, VALUE_STR, INDEX$_VALERR);
;  867        U 1286  2
;  868        U 1287  2          CMDBLK [NDX$H_LEVEL] = .QUALIFIER_VALUE - 1;
;  869        U 1288  2          END
;  870        U 1289  2      ELSE
;  871        U 1290  2          BEGIN
;  872        U 1291  2
;  873        U 1292  2          IF  .CMDBLK [NDX$V_MASTER]
;  874        U 1293  2          THEN
;  875        U 1294  2              CMDBLK [NDX$H_LEVEL] = 1;           ! Levels 0 and 1 for /MASTER
;  876        U 1295  2
;  877        U 1296  2          END;
;  878        U 1297  2
;  879          1298  2  %FI
;  880          1299  2
;  881          1300  2      !
;  882          1301  2      ! /[NO]GUIDE_HEADINGS
;  883          1302  2      !
;  884          1303  2      CMDBLK [NDX$V_GUIDE] = FALSE;
;  885          1304  2
;  886        L 1305  2  %IF DSRPLUS
;  887        U 1306  2  %THEN
;  888        U 1307  2
;  889        U 1308  2      IF  CLI$PRESENT (%ASCID'GUIDE_HEADINGS')
;  890        U 1309  2      THEN
;  891        U 1310  2          CMDBLK [NDX$V_GUIDE] = TRUE;
;  892        U 1311  2
```

```
;  893          1312  2 %FI
;  894          1313  2
;  895          1314  2     !
;  896          1315  2     ! /[NO]IDENTIFICATION
;  897          1316  2     !
;  898          1317  2
;  899          1318  2     IF  CLI$PRESENT (%ASCID'IDENTIFICATION')
;  900          1319  2     THEN
;  901          1320  2         SIGNAL (INDEX$_IDENT, 2, .NDXVRL, .NDXVRP);
;  902          1321  2
;  903          1322  2     !
;  904          1323  2     ! /[NO]LOG
;  905          1324  2     !
;  906          1325  2
;  907          1326  2     IF  CLI$PRESENT (%ASCID'LOG')
;  908          1327  2     THEN
;  909          1328  2         CMDBLK [NDX$V_LOG] = TRUE
;  910          1329  2     ELSE
;  911          1330  2         CMDBLK [NDX$V_LOG] = FALSE;
;  912          1331  2
;  913          1332  2     !
;  914          1333  2     ! /NOOUTPUT
;  915          1334  2     ! /OUTPUT = filespec
;  916          1335  2     !
;  917          1336  2
;  918          1337  2     IF CLI$PRESENT (%ASCID'OUTPUT')
;  919          1338  2     THEN
;  920          1339  3         BEGIN
;  921          1340  3         CMDBLK [NDX$V_OUTPUT] = TRUE;
;  922          1341  3
;  923          1342  3         CLI$GET_VALUE (%ASCID'OUTPUT', CMDBLK [NDX$T_OUTPUT_FILE]);
;  924          1343  3         END
;  925          1344  2     ELSE
;  926          1345  2         CMDBLK [NDX$V_OUTPUT] = FALSE;
;  927          1346  2
;  928          1347  2     !
;  929          1348  2     ! /[NO]OVERRIDE
;  930          1349  2     !
;  931          1350  2     CMDBLK [NDX$V_OVERRIDE] = FALSE;
;  932          1351  2
;  933        L 1352  2 %IF DSRPLUS
;  934        U 1353  2 %THEN
;  935        U 1354  2
;  936        U 1355  2     IF  CLI$PRESENT (%ASCID'OVERRIDE_MASTER')
;  937        U 1356  2     THEN
;  938        U 1357  2         CMDBLK [NDX$V_OVERRIDE] = TRUE;
;  939        U 1358  2
;  940          1359  2 %FI
;  941          1360  2
;  942          1361  2     !
;  943          1362  2     ! /NOPAGE_NUMBERS
;  944          1363  2     ! /PAGE_NUMBERS = ([[NO]RUNNING], [[NO]MERGE])
;  945          1364  2     !
;  946          1365  2     ! NORUNNING is the same as STANDARD.
;  947          1366  2     !
;  948          1367  2     CMDBLK [NDX$V_STANDARD_PAGE] = TRUE;        ! Generate standard page numbers
;  949          1368  2     CMDBLK [NDX$V_PAGES]         = TRUE;        ! Generate page numbers
```

```
;  950          1369  2
;  951        L 1370  2 %IF NOT DSRPLUS
;  952          1371  2 %THEN
;  953          1372  2
;  954          1373  2     CMDBLK [NDX$V_PAGE_MERGE]    = TRUE;        ! Merge page numbers for DSR
;  955          1374  2
;  956        U 1375  2 %ELSE
;  957        U 1376  2
;  958        U 1377  2     CMDBLK [NDX$V_PAGE_MERGE]    = FALSE;       ! Page ranges formed by .XPLUS (BEGIN - END)
;  959        U 1378  2
;  960          1379  2 %FI
;  961          1380  2
;  962          1381  2     SELECTONE CLI$PRESENT (%ASCID'PAGE_NUMBERS') OF
;  963          1382  2         SET
;  964          1383  2
;  965          1384  2         [CLI$_NEGATED]:
;  966          1385  2             !
;  967          1386  2             ! Qualifier explicitly negated (/NOPAGE_NUMBERS).
;  968          1387  2             !
;  969          1388  2             CMDBLK [NDX$V_PAGES] = FALSE;
;  970          1389  2
;  971          1390  2         [CLI$_PRESENT]:
;  972          1391  2             BEGIN
;  973          1392  2             !
;  974          1393  2             ! Qualifier was given explicitly on command line.
;  975          1394  2             !
;  976          1395  2             WHILE CLI$GET_VALUE (%ASCID'PAGE_NUMBERS', VALUE_STR) DO
;  977          1396  2
;  978          1397  2                 IF  NOT CALL_TPARSE (VALUE_STR, PAGE_STATE, PAGE_KEY)
;  979          1398  2                 THEN
;  980          1399  2                     SIGNAL_STOP (INDEX$_BADVALUE, 1, VALUE_STR);
;  981          1400  2
;  982          1401  2             END;
;  983          1402  2
;  984          1403  2         [OTHERWISE]:
;  985          1404  2             !
;  986          1405  2             ! CLI$_ABSENT, CLI$_DEFAULTED.
;  987          1406  2             ! Qualifier is present by default.
;  988          1407  2             !
;  989          1408  2             ;
;  990          1409  2
;  991          1410  2         TES;
;  992          1411  2
;  993          1412  2     !
;  994          1413  2     ! /NOREQUIRE
;  995          1414  2     ! /REQUIRE = filespec
;  996          1415  2     !
;  997          1416  2     CMDBLK [NDX$V_REQUIRE] = FALSE;
;  998          1417  2
;  999          1418  2     IF  CLI$PRESENT (%ASCID'REQUIRE')
; 1000          1419  2     THEN
; 1001          1420  2         BEGIN
; 1002          1421  2         CMDBLK [NDX$V_REQUIRE] = TRUE;
; 1003          1422  2         CLI$GET_VALUE (%ASCID'REQUIRE', CMDBLK [NDX$T_REQUIRE_FILE]);
; 1004          1423  2         END;
; 1005          1424  2
; 1006          1425  2     !
```

```
;  1007          1426  2     ! /SORT = ([{ WORD ! LETTER }], [NONALPHA = { IGNORE ! BEFORE ! AFTER }])
;  1008          1427  2     !
;  1009          1428  2     CMDBLK [NDX$V_WORD_SORT] = TRUE;            ! Word by word sort is default
;  1010          1429  2     CMDBLK [NDX$H_NONA[PHA]  = IGNORE;          ! Ignore leading nonalphas
;  1011          1430  2
;  1012        L 1431  2 %IF DSRPLUS
;  1013        U 1432  2 %THEN
;  1014        U 1433  2
;  1015        U 1434  2     IF  CLI$PRESENT (%ASCID'SORT')
;  1016        U 1435  2     THEN
;  1017        U 1436  2
;  1018        U 1437  2         WHILE CLI$GET_VALUE (%ASCID'SORT', VALUE_STR) DO
;  1019        U 1438  2
;  1020        U 1439  2             IF  NOT CALL_TPARSE (VALUE_STR, SORT_STATE, SORT_KEY)
;  1021        U 1440  2             THEN
;  1022        U 1441  2                 SIGNAL_STOP (INDEX$_BADVALUE, 1, VALUE_STR);
;  1023        U 1442  2
;  1024          1443  2 %FI
;  1025          1444  2
;  1026          1445  2     !
;  1027          1446  2     ! Process all input files and local qualifiers
;  1028          1447  2     !
;  1029          1448  2     CMDBLK [NDX$V_INPUT_CONCAT] = FALSE;
;  1030          1449  2
;  1031          1450  2     WHILE (STATUS = CLI$GET_VALUE (%ASCID'INPUT', CMDBLK [NDX$T_INPUT_FILE])) DO
;  1032          1451  2         BEGIN
;  1033          1452  2
;  1034        L 1453  2 %IF DSRPLUS
;  1035        U 1454  2 %THEN
;  1036        U 1455  2
;  1037        U 1456  2         !
;  1038        U 1457  2         ! /OPTIONS - input file is an options file
;  1039        U 1458  2         !
;  1040        U 1459  2         IF  CLI$PRESENT (%ASCID'OPTIONS')
;  1041        U 1460  2         THEN
;  1042        U 1461  2             BEGIN
;  1043        U 1462  2             !
;  1044        U 1463  2             ! Make sure /BOOK_IDENTIFIER was not also specified.
;  1045        U 1464  2             ! Make sure options file is last in concatenated list.
;  1046        U 1465  2             !
;  1047        U 1466  2
;  1048        U 1467  2             IF  CLI$PRESENT (%ASCID'BOOK_IDENTIFIER')
;  1049        U 1468  2             THEN
;  1050        U 1469  2                 SIGNAL (INDEX$_IGNORED, 1, %ASCID'BOOK_IDENTIFIER', INDEX$_CONFQUAL);
;  1051        U 1470  2
;  1052        U 1471  2             IF .STATUS EQL CLI$_CONCAT
;  1053        U 1472  2             THEN
;  1054        U 1473  2                 BEGIN
;  1055        U 1474  2                 !
;  1056        U 1475  2                 ! Current input file concatenated to next - error.
;  1057        U 1476  2                 !
;  1058        U 1477  2                 CLI$GET_VALUE (%ASCID'INPUT', VALUE_STR);
;  1059        U 1478  2                 SIGNAL (INDEX$_IGNORED, 1, VALUE_STR, INDEX$_NOLIST);
;  1060        U 1479  2                 END;
;  1061        U 1480  2
;  1062        U 1481  2             !
;  1063        U 1482  2             ! Process options file and exit loop
```

```
; 1064        U 1483  3               !
; 1065        U 1484  3               OPTIONS_FILE ();
; 1066        U 1485  3               EXITLOOP;
; 1067        U 1486  3               END;
; 1068        U 1487  3
; 1069        U 1488  3           !
; 1070        U 1489  3           ! Process /BOOK_IDENTIFIER qualifier if present
; 1071        U 1490  3           !
; 1072        U 1491  3
; 1073        U 1492  3           PARSE_BOOK ();
; 1074        U 1493  3
; 1075          1494  3       %FI
; 1076          1495  3
; 1077          1496  3           !
; 1078          1497  3           ! Process this input file.
; 1079          1498  3           !
; 1080          1499  3           ! NDXINP () will call MAKNDX to generate an output index
; 1081          1500  3           ! if this input file is not concatenated to the previous one.
; 1082          1501  3           !
; 1083          1502  3           NDXINP ();
; 1084          1503  3
; 1085          1504  3           CMDBLK [NDX$V_INPUT_CONCAT] = (.STATUS EQL CLI$_CONCAT);
; 1086          1505  3           END;
; 1087          1506  3
; 1088          1507  2       !
; 1089          1508  2       ! Generate last output index and clean up
; 1090          1509  2       !
; 1091          1510  2       MAKNDX ();
; 1092          1511  2
; 1093          1512  2       RETURN (.TERMINATION_STATUS OR STS$M_INHIB_MSG);
; 1094          1513  1       END;


;                                                       .TITLE  NDXVMS NDXVMS -- DSRINDEX/INDEX Command line in
                                                                terface
                                                        .IDENT  \V04-000\

                                                        .PSECT  _LIB$KEY1$,NOWRT,  SHR,  PIC,1

                                                00000 ;TPA$KEYSTO
                                                U.9:    .BLKB   0
                                47  4E  49  4E  4E  55  52 00000 ;TPA$KEYST
                                                U.11:   .ASCII  \RUNNING\                               ;
                                                00007           .BYTE   -1                              ;
                                                00008 ;TPA$KEYSTO
                                                U.16:   .BLKB   0
                            47  4E  49  4E  4E  55  52  4F  4E 00008 ;TPA$KEYST
                                                U.18:   .ASCII  \NORUNNING\                             ;
                                                00011           .BYTE   -1                              ;
                                                00012 ;TPA$KEYFILL
                                                U.23:   .BYTE   -1                                      ;

                                                        .PSECT  _LIB$STATE$,NOWRT,  SHR,  PIC,1

                                                00000 NUMBER_STATE::
                                                        .BLKB   0
                                            41F3 00000 ;TPA$TYPE
```

```
                                                U.2:    .WORD   16883                   ;
                                00000000* 00002 ;TPA$ADDR
                                                U.3:    .LONG   <<QUALIFIER_VALUE-U.3>-4>       ;
                                    15F7  00006 ;TPA$TYPE
                                                U.4:    .WORD   5623                    ;
                                    FFFF  00008 ;TPA$TARGET
                                                U.5:    .WORD   -1                      ;
                                    15F7  0000A ;TPA$TYPE
                                                U.6:    .WORD   5623                    ;
                                    FFFF  0000C ;TPA$TARGET
                                                U.7:    .WORD   -1                      ;
                                          0000E         .BLKB   2
                                          00010 PAGE_STATE::
                                                        .BLKB   0
                                    8300  00010 ;TPA$TYPE
                                                U.12:   .WORD   -32000                  ;
                                      01  00012 ;TPA$FLAGS2
                                                U.13:   .BYTE   1                       ;
                                00000000  00013 ;TPA$PARAM
                                                U.14:   .LONG   0                       ;
                                00000000V 00017 ;TPA$ACTION
                                                U.15:   .LONG   <<ENTER_PAGE-U.15>-4>   ;
                                    8701  0001B ;TPA$TYPE
                                                U.19:   .WORD   -30975                  ;
                                      01  0001D ;TPA$FLAGS2
                                                U.20:   .BYTE   1                       ;
                                00000001  0001E ;TPA$PARAM
                                                U.21:   .LONG   1                       ;
                                00000000V 00022 ;TPA$ACTION
                                                U.22:   .LONG   <<ENTER_PAGE-U.22>-4>   ;
                                    15F7  00026 ;TPA$TYPE
                                                U.24:   .WORD   5623                    ;
                                    FFFF  00028 ;TPA$TARGET
                                                U.25:   .WORD   -1                      ;

                                                        .PSECT  _LIB$KEY0$,NOWRT,  SHR,  PIC,1

                                          00000 NUMBER_KEY::
                                                        .BLKB   0
                                          00000 ;TPA$KEY0
                                                U.1:    .BLKB   0
                                          00000 PAGE_KEY::
                                                        .BLKB   0
                                          00000 ;TPA$KEY0
                                                U.8:    .BLKB   0
                                    0000* 00000 ;TPA$KEY
                                                U.10:   .WORD   <U.9-U.8>                ;
                                    0000* 00002 ;TPA$KEY
                                                U.17:   .WORD   <U.16-U.8>               ;

                                                        .PSECT  $PLIT$,NOWRT,NOEXE,2

                      00  00  00  45  4E  49  4C  24  00000 P.AAB:   .ASCII  \$LINE\<0><0><0>
                                        010E0005  00008 P.AAA:   .LONG   17694725
                                        00000000' 0000C         .ADDRESS P.AAB
00  45  47  41  50  5F  52  45  50  5F  53  45  4E  49  4C  00010 P.AAD:   .ASCII  \LINES_PER_PAGE\<0><0>
                                              00  0001F
```

```
                                                    010E000E  00020 P.AAC:   .LONG   17694734
                                                    00000000' 00024        .ADDRESS P.AAD
00  45  47  41  50  5F  52  45  50  5F  53  45  4E  49  4C  00028 P.AAF:   .ASCII  \LINES_PER_PAGE\<0><0>
                                                        00  00037
                                                    010E000E  00038 P.AAE:   .LONG   17694734
                                                    00000000' 0003C        .ADDRESS P.AAF
00  45  47  41  50  5F  52  45  50  5F  53  45  4E  49  4C  00040 P.AAH:   .ASCII  \LINES_PER_PAGE\<0><0>
                                                        00  0004F
                                                    010E000E  00050 P.AAG:   .LONG   17694734
                                                    00000000' 00054        .ADDRESS P.AAH
                                    00  45  56  52  45  53  45  52  00058 P.AAJ:   .ASCII  \RESERVE\<0>
                                                    010E0007  00060 P.AAI:   .LONG   17694727
                                                    00000000' 00064        .ADDRESS P.AAJ
                                    00  45  56  52  45  53  45  52  00068 P.AAL:   .ASCII  \RESERVE\<0>
                                                    010E0007  00070 P.AAK:   .LONG   17694727
                                                    00000000' 00074        .ADDRESS P.AAL
00  4E  4F  49  54  41  43  49  46  49  54  4E  45  44  49  00078 P.AAN:   .ASCII  \IDENTIFICATION\<0><0>
                                                        00  00087
                                                    010E000E  00088 P.AAM:   .LONG   17694734
                                                    00000000' 0008C        .ADDRESS P.AAN
                                            00  47  4F  4C  00090 P.AAP:   .ASCII  \LOG\<0>
                                                    010E0003  00094 P.AAO:   .LONG   17694723
                                                    00000000' 00098        .ADDRESS P.AAP
                                    00  00  54  55  50  54  55  4F  0009C P.AAR:   .ASCII  \OUTPUT\<0><0>
                                                    010E0006  000A4 P.AAQ:   .LONG   17694726
                                                    00000000' 000A8        .ADDRESS P.AAR
                                    00  00  54  55  50  54  55  4F  000AC P.AAT:   .ASCII  \OUTPUT\<0><0>
                                                    010E0006  000B4 P.AAS:   .LONG   17694726
                                                    00000000' 000B8        .ADDRESS P.AAT
53  52  45  42  4D  55  4E  5F  45  47  41  50      000BC P.AAV:   .ASCII  \PAGE_NUMBERS\
                                                    010E000C  000C8 P.AAU:   .LONG   17694732
                                                    00000000' 000CC        .ADDRESS P.AAV
53  52  45  42  4D  55  4E  5F  45  47  41  50      000D0 P.AAX:   .ASCII  \PAGE_NUMBERS\
                                                    010E000C  000DC P.AAW:   .LONG   17694732
                                                    00000000' 000E0        .ADDRESS P.AAX
                                00  45  52  49  55  51  45  52  000E4 P.AAZ:   .ASCII  \REQUIRE\<0>
                                                    010E0007  000EC P.AAY:   .LONG   17694727
                                                    00000000' 000F0        .ADDRESS P.AAZ
                                00  45  52  49  55  51  45  52  000F4 P.ABB:   .ASCII  \REQUIRE\<0>
                                                    010E0007  000FC P.ABA:   .LONG   17694727
                                                    00000000' 00100        .ADDRESS P.ABB
                                00  00  00  54  55  50  4E  49  00104 P.ABD:   .ASCII  \INPUT\<0><0><0>
                                                    010E0005  0010C P.ABC:   .LONG   17694725
                                                    00000000' 00110        .ADDRESS P.ABD

                                                                           .PSECT  $OWN$,NOEXE,2

                                                        0000  00000 VALUE_STR:
                                                                           .WORD   0
                                                          02  0E  00002           .BYTE   14, 2
                                                    00000000  00004           .LONG   0
                                                        0000  00008 OPTIONS_STR:
                                                                           .WORD   0
                                                          02  0E  0000A           .BYTE   14, 2
                                                    00000000  0000C           .LONG   0
                                                              00010 QUALIFIER_VALUE:
                                                                           .BLKB   4
```

```
                                                     ; 14
                                00000001  00014 TERMINATION_STATUS:
                                                        .LONG   1                                                       ;

                                                        .EXTRN  DSRINDEX$_BADLOGIC
                                                        .EXTRN  DSRINDEX$_BADVALUE
                                                        .EXTRN  DSRINDEX$_INSVIRMEM
                                                        .EXTRN  DSRINDEX$_LINELENG
                                                        .EXTRN  DSRINDEX$_NOREF
                                                        .EXTRN  DSRINDEX$_OPENIN
                                                        .EXTRN  DSRINDEX$_OPENOUT
                                                        .EXTRN  DSRINDEX$_TOOMANY
                                                        .EXTRN  DSRINDEX$_VALERR
                                                        .EXTRN  DSRINDEX$_CANTBAL
                                                        .EXTRN  DSRINDEX$_CLOSEQUOT
                                                        .EXTRN  DSRINDEX$_CONFQUAL
                                                        .EXTRN  DSRINDEX$_CTRLCHAR
                                                        .EXTRN  DSRINDEX$_DOESNTFIT
                                                        .EXTRN  DSRINDEX$_DUPBEGIN
                                                        .EXTRN  DSRINDEX$_EMPTYIN
                                                        .EXTRN  DSRINDEX$_IGNORED
                                                        .EXTRN  DSRINDEX$_INVINPUT
                                                        .EXTRN  DSRINDEX$_INVRECORD
                                                        .EXTRN  DSRINDEX$_LASTCONT
                                                        .EXTRN  DSRINDEX$_NOBEGIN
                                                        .EXTRN  DSRINDEX$_NOEND
                                                        .EXTRN  DSRINDEX$_NOINDEX
                                                        .EXTRN  DSRINDEX$_NOLIST
                                                        .EXTRN  DSRINDEX$_OVERSTRK
                                                        .EXTRN  DSRINDEX$_SKIPPED
                                                        .EXTRN  DSRINDEX$_SYNTAX
                                                        .EXTRN  DSRINDEX$_TEXFILE
                                                        .EXTRN  DSRINDEX$_TOODEEP
                                                        .EXTRN  DSRINDEX$_TOOFEW
                                                        .EXTRN  DSRINDEX$_TRUNCATED
                                                        .EXTRN  DSRINDEX$_COMPLETE
                                                        .EXTRN  DSRINDEX$_CREATED
                                                        .EXTRN  DSRINDEX$_IDENT
                                                        .EXTRN  DSRINDEX$_PROCFILE
                                                        .EXTRN  DSRINDEX$_TEXT, DSRINDEX$_TEXTD
                                                        .EXTRN  DSRINDEX$_TMS11
                                                        .EXTRN  TAB, TMSCOL, MAXLIN
                                                        .EXTRN  CLI$_CONCAT, CLI$_PRESENT
                                                        .EXTRN  CLI$_NEGATED, CLI$_DEFAULTED
                                                        .EXTRN  CLI$_ABSENT, CMDBLK
                                                        .EXTRN  CHRSIZ, CHRSZA, CHRSZE
                                                        .EXTRN  NDXVRL, NDXVRP, NDXINI
                                                        .EXTRN  NDXINP, MAKNDX, CLI$PRESENT
                                                        .EXTRN  CLI$GET_VALUE, LIB$TPARSE

                                                        .PSECT  $CODE$,NOWRT,2

                                    0FFC 00000          .ENTRY  NDXCLI, Save R2,R3,R4,R5,R6,R7,R8,R9,R10,-              ; 0800
                                                                R11
5B 00000000'   EF   9E 00002        MOVAB   NUMBER_STATE, R11                                                           ;
5A 00000000'   EF   9E 00009        MOVAB   NUMBER_KEY, R10                                                             ;
59 00000000G   8F   D0 00010        MOVL    #DSRINDEX$_BADVALUE, R9                                                     ;
58 00000000G   00   9E 00017        MOVAB   LIB$STOP, R8                                                                ;
```

```
                        57 00000000G   00  9E 0001E          MOVAB   CLI$PRESENT, R7                                  ;
                        56 00000000G   00  9E 00025          MOVAB   CLI$GET_VALUE, R6                                ;
                        55 00000000'   EF  9E 0002C          MOVAB   P.AAA, R5                                        ;
                        54 00000000'   EF  9E 00033          MOVAB   VALUE_STR, R4                                    ;
                        53 00000000G   EF  9E 0003A          MOVAB   CMDBLK, R3                                       ;
                        6D      0272   CF  DE 00041          MOVAL   30$, (FP)                                        ; 0832
              00000000G EF             00  FB 00046          CALLS   #0, NDXINI                                       ; 0840
                              48       A3  9F 0004D          PUSHAB  CMDBLK+72                                        ; 0845
                                       55  DD 00050          PUSHL   R5                                               ;
                        66             02  FB 00052          CALLS   #2, CLI$GET_VALUE                                ;
                     01 A3             04  8A 00055          BICB2   #4, CMDBLK+1                                     ; 0857
              00000000G EF 00000000G   EF  9E 00059          MOVAB   CHRSZA, CHRSIZ                                   ; 0878
                     0C A3             22  D0 00064          MOVL    #34, CMDBLK+12                                   ; 0908
                     C4 A3 00010001    8F  D0 00068          MOVL    #65537, CMDBLK+4                                 ; 0877
                     1C A3       0C    A3  D0 00070          MOVL    CMDBLK+12, CMDBLK+28                             ; 0956
                     01 A3             10  8A 00075          BICB2   #16, CMDBLK+1                                    ; 1023
                        01       04    A3  B1 00079          CMPW    CMDBLK+4, #1                                     ; 1056
                                       11  12 0007D          BNEQ    2$                                               ;
                  06    01 A3          04  E1 0007F          BBC     #4, CMDBLK+1, 1$                                 ; 1060
                        14 A3          34  D0 00084          MOVL    #52, CMDBLK+20                                   ; 1062
                                       0A  11 00088          BRB     3$                                               ;
                        14 A3          37  D0 0008A 1$:      MOVL    #55, CMDBLK+20                                   ; 1064
                                       04  11 0008E          BRB     3$                                               ; 1056
                        14 A3          36  D0 00090 2$:      MOVL    #54, CMDBLK+20                                   ; 1068
                                 18    A5  9F 00094 3$:      PUSHAB  P.AAC                                            ; 1070
                        67             01  FB 00097          CALLS   #1, CLI$PRESENT                                  ;
                        7B             50  E9 0009A          BLBC    R0, 9$                                           ;
                        04       06    A3  B1 0009D          CMPW    CMDBLK+6, #4                                     ; 1077
                                       1A  12 000A1          BNEQ    4$                                               ;
                           00000000G   8F  DD 000A3          PUSHL   #DSRINDEX$_CONFQUAL                              ; 1082
                                 30    A5  9F 000A9          PUSHAB  P.AAE                                            ;
                                       01  DD 000AC          PUSHL   #1                                               ;
                           00000000G   8F  DD 000AE          PUSHL   #DSRINDEX$_IGNORED                               ;
              00000000G 00             04  FB 000B4          CALLS   #4, LIB$SIGNAL                                   ;
                                       5B  11 000BB          BRB     9$                                               ;
                              10       A4  D4 000BD 4$:      CLRL    QUALIFIER_VALUE                                  ; 1085
                                       54  DD 000C0          PUSHL   R4                                               ; 1086
                              48       A5  9F 000C2          PUSHAB  P.AAG                                            ;
                        66             02  FB 000C5          CALLS   #2, CLI$GET_VALUE                                ;
                                       5A  DD 000C8          PUSHL   R10                                              ; 1088
                                 0810  8F  BB 000CA          PUSHR   #^M<R4,R11>                                      ;
              00000000V EF             03  FB 000CE          CALLS   #3, CALL_TPARSE                                  ;
                        09             50  E8 000D5          BLBS    R0, 5$                                           ;
                                       54  DD 000D8          PUSHL   R4                                               ; 1090
                                       01  DD 000DA          PUSHL   #1                                               ;
                                       59  DD 000DC          PUSHL   R9                                               ;
                        68             03  FB 000DE          CALLS   #3, LIB$STOP                                     ;
                     14 A3       10    A4  D0 000E1 5$:      MOVL    QUALIFIER_VALUE, CMDBLK+20                       ; 1092
                        50       14    A3  D0 000E6          MOVL    CMDBLK+20, R0                                    ; 1095
                        0F             50  D1 000EA          CMPL    R0, #15                                          ;
                                       06  18 000ED          BGEQ    6$                                               ;
                        01       04    A3  B1 000EF          CMPW    CMDBLK+4, #1                                     ; 1096
                                       14  13 000F3          BEQL    8$                                               ;
                        19             50  D1 000F5 6$:      CMPL    R0, #25                                          ; 1099
                                       06  18 000F8          BGEQ    7$                                               ;
                        01       04    A3  B1 000FA          CMPW    CMDBLK+4, #1                                     ; 1100
                                       09  12 000FE          BNEQ    8$                                               ;
```

```
                00000000G  8F              50  D1 00100 7$:     CMPL    R0, #MAXLIN                          ; 1102
                                            0F  15 00107         BLEQ    9$
                           00000000G  8F    DD 00109 8$:     PUSHL   #DSRINDEX$_VALERR                    ; 1104
                                      54    DD 0010F         PUSHL   R4
                                      01    DD 00111         PUSHL   #1
                                      59    DD 00113         PUSHL   R9
                           68                 04  FB 00115         CALLS   #4, LIB$STOP
                      10   A3                 02  D0 00118 9$:     MOVL    #2, CMDBLK+16                ; 1119
                           51        06   A3  32 0011C         CVTWL   CMDBLK+6, R1                 ; 1173
                           01             51  B1 00120         CMPW    R1, #1                       ; 1176
                                          0B  12 00123         BNEQ    10$
                           50        0C   A3  D0 00125         MOVL    CMDBLK+12, R0                ; 1178
                           50        10 B340  3E 00129         MOVAW   @CMDBLK+16[R0], R0
                                          0F  11 0012E         BRB     11$                          ; 1179
                           03             51  B1 00130 10$:     CMPW    R1, #3                       ; 1183
                                          13  12 00133         BNEQ    12$
                50         0C   A3        10  A3 C1 00135         ADDL3   CMDBLK+16, CMDBLK+12, R0     ; 1186
                           50        1C   A3  C0 0013B         ADDL2   CMDBLK+28, R0                ; 1187
                      00000078   8F          50  D1 0013F 11$:     CMPL    R0, #120
                                          08  11 00146         BRB     13$
                      00000078   8F        0C   A3  D1 00148 12$:     CMPL    CMDBLK+12, #120              ; 1193
                                          09  15 00150 13$:     BLEQ    14$
                           00000000G      8F  DD 00152         PUSHL   #DSRINDEX$_LINELENG          ; 1195
                           68                 01  FB 00158         CALLS   #1, LIB$STOP
                           63        40   8F  8A 0015B 14$:     BICB2   #64, CMDBLK                  ; 1208
                                     18   A3  D4 0015F         CLRL    CMDBLK+24                    ; 1241
                                     58   A5  9F 00162         PUSHAB  P.AAI                        ; 1243
                           67                 01  FB 00165         CALLS   #1, CLI$PRESENT
                           3F                 50  E9 00168         BLBC    R0, 17$
                                     10   A4  D4 0016B         CLRL    QUALIFIER_VALUE              ; 1246
                                     54  DD 0016E         PUSHL   R4                           ; 1247
                                     68  A5  9F 00170         PUSHAB  P.AAK
                           66                 02  FB 00173         CALLS   #2, CLI$GET_VALUE
                                     5A  DD 00176         PUSHL   R10                          ; 1249
                                0810     8F  BB 00178         PUSHR   #^M<R4,R11>
                00000000V  EF                 03  FB 0017C         CALLS   #3, CALL_TPARSE
                           09                 50  E8 00183         BLBS    R0, 15$
                                     54  DD 00186         PUSHL   R4                           ; 1251
                                     01  DD 00188         PUSHL   #1
                                     59  DD 0018A         PUSHL   R9
                           68                 03  FB 0018C         CALLS   #3, LIB$STOP
                      14   A3        10   A4  D1 0018F 15$:     CMPL    QUALIFIER_VALUE, CMDBLK+20   ; 1253
                                          0F  15 00194         BLEQ    16$
                                00000000G  8F  DD 00196         PUSHL   #DSRINDEX$_VALERR            ; 1255
                                     54  DD 0019C         PUSHL   R4
                                     01  DD 0019E         PUSHL   #1
                                     59  DD 001A0         PUSHL   R9
                           68                 04  FB 001A2         CALLS   #4, LIB$STOP
                      18   A3        10   A4  D0 001A5 16$:     MOVL    QUALIFIER_VALUE, CMDBLK+24   ; 1257
                      0A   A3        63   8F  9B 001AA 17$:     MOVZBW  #99, CMDBLK+10               ; 1269
                           63        80   8F  8A 001AF         BICB2   #128, CMDBLK                 ; 1303
                                0080     C5  9F 001B3         PUSHAB  P.AAM                        ; 1318
                           67                 01  FB 001B7         CALLS   #1, CLI$PRESENT
                           18                 50  E9 001BA         BLBC    R0, 18$
                                00000000G  EF  DD 001BD         PUSHL   NDXVRP                       ; 1320
                                00000000G  EF  DD 001C3         PUSHL   NDXVRL
                                           02  DD 001C9         PUSHL   #2
```

```
                                00000000G  8F   DD 001CB           PUSHL   #DSRINDEX$_IDENT
                        00000000G   00         04 FB 001D1        CALLS   #4, LIB$SIGNAL
                                      008C   C5 9F 001D8 18$:     PUSHAB  P.AAO                   1326
                                67            01 FB 001DC        CALLS   #1, CLI$PRESENT
                                06            50 E9 001DF        BLBC    R0, 19$
                             01 A3            02 88 001E2        BISB2   #2, CMDBLK+1            1328
                                              04 11 001E6        BRB     20$
                             01 A3            02 8A 001E8 19$:   BICB2   #2, CMDBLK+1            1330
                                      009C   C5 9F 001EC 20$:     PUSHAB  P.AAQ                   1337
                                67            01 FB 001F0        CALLS   #1, CLI$PRESENT
                                0F            50 E9 001F3        BLBC    R0, 21$
                                63            02 88 001F6        BISB2   #2, CMDBLK              1340
                                         30   A3 9F 001F9        PUSHAB  CMDBLK+48               1342
                                      00AC   C5 9F 001FC        PUSHAB  P.AAS
                                66            02 FB 00200        CALLS   #2, CLI$GET_VALUE
                                              03 11 00203        BRB     22$                     1337
                                63            02 8A 00205 21$:   BICB2   #2, CMDBLK              1345
                                63            10 8A 00208 22$:   BICB2   #16, CMDBLK             1350
                                63      0828  8F A8 0020B        BISW2   #2088, CMDBLK           1373
                                      00C0   C5 9F 00210        PUSHAB  P.AAU                   1381
                                67            01 FB 00214        CALLS   #1, CLI$PRESENT
                        00000000G   8F        50 D1 00217        CMPL    R0, #CLI$_NEGATED       1384
                                              05 12 0021E        BNEQ    23$
                                63            08 8A 00220        BICB2   #8, CMDBLK              1388
                                              31 11 00223        BRB     25$
                        00000000G   8F        50 D1 00225 23$:   CMPL    R0, #CLI$_PRESENT       1390
                                              28 12 0022C        BNEQ    25$
                                              54 DD 0022E 24$:   PUSHL   R4                      1395
                                      00D4   C5 9F 00230        PUSHAB  P.AAW
                                66            02 FB 00234        CALLS   #2, CLI$GET_VALUE
                                1C            50 E9 00237        BLBC    R0, 25$
                                              5A DD 0023A        PUSHL   R10                     1397
                                         10   AB 9F 0023C        PUSHAB  PAGE_STATE
                                              54 DD 0023F        PUSHL   R4
                        00000000V   EF        03 FB 00241        CALLS   #3, CALL_TPARSE
                                E3            50 E8 00248        BLBS    R0, 24$
                                              54 DD 0024B        PUSHL   R4                      1399
                                              01 DD 0024D        PUSHL   #1
                                              59 DD 0024F        PUSHL   R9
                                68            03 FB 00251        CALLS   #3, LIB$STOP
                                              D8 11 00254        BRB     24$                     1397
                                63            04 8A 00256 25$:   BICB2   #4, CMDBLK              1416
                                      00E4   C5 9F 00259        PUSHAB  P.AAY                   1418
                                67            01 FB 0025D        CALLS   #1, CLI$PRESENT
                                0D            50 E9 00260        BLBC    R0, 26$
                                63            04 88 00253        BISB2   #4, CMDBLK              1421
                                         38   A3 9F 00266        PUSHAB  CMDBLK+56               1422
                                      00F4   C5 9F 00269        PUSHAB  P.ABA
                                66            02 FB 0026D        CALLS   #2, CLI$GET_VALUE
                             01 A3            01 88 00270 26$:   BISB2   #1, CMDBLK+1            1428
                             08 A3            03 B0 00274        MOVW    #3, CMDBLK+8            1429
                                63            01 8A 00278        BICB2   #1, CMDBLK              1448
                                         28   A3 9F 0027B 27$:   PUSHAB  CMDBLK+40               1450
                                      0104   C5 9F 0027E        PUSHAB  P.ABC
                                66            02 FB 00282        CALLS   #2, CLI$GET_VALUE
                                52            50 D0 00285        MOVL    R0, STATUS
                                1B            52 E9 00288        BLBC    STATUS, 29$
```

```
                        00000000G  EF                00  FB  0028B          CALLS   #0, NDXINP                                  ; 1502
                                                     50  D4  00292          CLRL    R0                                          ; 1504
                        00000000G  8F                52  D1  00294          CMPL    STATUS, #CLI$_CONCAT
                                                     02  12  0029B          BNEQ    28$
                                                     50  D6  0029D          INCL    R0
            63             01          00            50  F0  0029F 28$:     INSV    R0, #0, #1, CMDBLK
                                                     D5  11  002A4          BRB     27$                                         ; 1450
                        00000000G  EF                00  FB  002A6 29$:     CALLS   #0, MAKNDX                                  ; 1510
                    50          14  A4 10000000      8F  C9  002AD          BISL3   #268435456, TERMINATION_STATUS, R0          ; 1512
                                                         04  002B6          RET                                                 ; 1513
                                                   0000  002B7 30$:     .WORD   Save nothing                                    ; 0832
                                                     7E  D4  002B9          CLRL    -(SP)
                                                     5E  DD  002BB          PUSHL   SP
                                7E          04       AC  7D  002BD          MOVQ    4(AP), -(SP)
                        00000000V  EF                03  FB  002C1          CALLS   #3, CONDITION_HANDLER
                                                         04  002C8          RET

; Routine Size:  713 bytes,    Routine Base:  $CODE$ + 0000
```

```
;  1096          1514  1 %SBTTL 'CONDITION_HANDLER - Main program condition handler - sets termination status'
;  1097          1515  1 ROUTINE CONDITION_HANDLER (SIG : REF BLOCK [, BYTE], MCH : REF BLOCK [, BYTE]) =
;  1098          1516  1 !++
;  1099          1517  1 !
;  1100          1518  1 ! FUNCTIONAL DESCRIPTION:
;  1101          1519  1 !
;  1102          1520  1 !       This routine is enabled by NDXCLI as a condition handler.
;  1103          1521  1 !       Whenever a signal is generated, the signal severity is examined.
;  1104          1522  1 !       If the condition is more severe than any previous condition,
;  1105          1523  1 !       (success, warning, error, severe error) the severity is recorded
;  1106          1524  1 !       in termination_status which is the condition severity. NDXCLI
;  1107          1525  1 !       returns the value of TERMINATION_STATUS as the program status
;  1108          1526  1 !       which will set the value of the DCL $STATUS variable.
;  1109          1527  1 !
;  1110          1528  1 ! FORMAL PARAMETERS:
;  1111          1529  1 !
;  1112          1530  1 !       SIG                     - address of signal array
;  1113          1531  1 !       MCH                     - address of mechanism array
;  1114          1532  1 !
;  1115          1533  1 ! IMPLICIT INPUTS:
;  1116          1534  1 !
;  1117          1535  1 !       TERMINATION_STATUS      - current termination severity
;  1118          1536  1 !
;  1119          1537  1 ! IMPLICIT OUTPUTS:
;  1120          1538  1 !
;  1121          1539  1 !       TERMINATION_STATUS      - may be set to the severity level in the
;  1122          1540  1 !                                 signalled condition if it is more severe
;  1123          1541  1 !
;  1124          1542  1 ! ROUTINE VALUE:
;  1125          1543  1 ! COMPLETION CODES:
;  1126          1544  1 !
;  1127          1545  1 !       SS$_RESIGNAL
;  1128          1546  1 !
;  1129          1547  1 ! SIDE EFFECTS:
;  1130          1548  1 !
;  1131          1549  1 !       None
;  1132          1550  1 !--
;  1133          1551  2     BEGIN
;  1134          1552  2
;  1135          1553  2     BIND
;  1136          1554  2         SIGNALLED_CONDITION = SIG [CHF$L_SIG_NAME] : BLOCK [, BYTE];
;  1137          1555  2
;  1138          1556  2     SELECTONE .SIGNALLED_CONDITION [STS$V_SEVERITY] OF
;  1139          1557  2         SET
;  1140          1558  2
;  1141          1559  2         [STS$K_WARNING]:
;  1142          1560  2             IF .TERMINATION_STATUS EQL STS$K_SUCCESS
;  1143          1561  2             THEN
;  1144          1562  2                 !
;  1145          1563  2                 ! A warning changes the termination status only if it was
;  1146          1564  2                 ! 'success' previously.
;  1147          1565  2                 !
;  1148          1566  2                 TERMINATION_STATUS = STS$K_WARNING;
;  1149          1567  2
;  1150          1568  2         [STS$K_ERROR]:
;  1151          1569  2             IF .TERMINATION_STATUS LSS STS$K_ERROR
;  1152          1570  2             THEN
```

```
;  1153          1571  2                    !
;  1154          1572  2                    ! An error status changes the termination status only if it
;  1155          1573  2                    ! was 'success' or 'warning' previously.
;  1156          1574  2                    !
;  1157          1575  2                    TERMINATION_STATUS = STS$K_ERROR;
;  1158          1576  2
;  1159          1577  2            [STS$K_SEVERE]:                         ! Severe error
;  1160          1578  2                TERMINATION_STATUS = STS$K_SEVERE;  ! set the termination status
;  1161          1579  2
;  1162          1580  2            [OTHERWISE]:                            ! Success or Informational
;  1163          1581  2                ;                                   ! Do nothing
;  1164          1582  2
;  1165          1583  2            TES;
;  1166          1584  2
;  1167          1585  2        RETURN SS$_RESIGNAL;                        ! Continue processing condition
;  1168          1586  1        END;


                                        0004 00000 CONDITION_HANDLER:
                                                   .WORD   Save R2                                                  ; 1515
                          52 00000000'  EF  9E 00002       MOVAB   TERMINATION_STATUS, R2
                    50        04        AC  C1 00009       ADDL3   #4, SIG, R0                                     ; 1554
                                        07  93 0000E       BITB    (R0), #7                                        ; 1559
                                            12 00011       BNEQ    1$
                                        01  D1 00013       CMPL    TERMINATION_STATUS, #1                          ; 1560
                                            12 00016       BNEQ    3$
                                            D4 00018       CLRL    TERMINATION_STATUS                              ; 1566
                                            11 0001A       BRB     3$                                              ; 1560
                02        60              03 ED 0001C 1$:  CMPZV   #0, #3, (R0), #2                               ; 1568
                                            12 00021       BNEQ    2$
                                        02  D1 00023       CMPL    TERMINATION_STATUS, #2                          ; 1569
                                            18 00026       BGEQ    3$
                                        62  D0 00028       MOVL    #2, TERMINATION_STATUS                          ; 1575
                                            11 0002B       BRB     3$                                              ; 1569
                04        60              03 ED 0002D 2$:  CMPZV   #0, #3, (R0), #4                               ; 1577
                                            12 00032       BNEQ    3$
                                        62  D0 00034       MOVL    #4, TERMINATION_STATUS                          ; 1578
                                        50  3C 00037 3$:    MOVZWL  #2328, R0                                       ; 1585
                                            04 0003C       RET                                                     ; 1586

; Routine Size:  61 bytes,    Routine Base:  $CODE$ + 02C9
```

```
;  1170          1587  1 %SBTTL 'CALL_TPARSE -- Invoke TPARSE to process qualifier values'
;  1171          1588  1 ROUTINE CALL_TPARSE (STRING : REF $STR_DESCRIPTOR (), STATE_TAB, KEY_TAB) =
;  1172          1589  1 !++
;  1173          1590  1 !
;  1174          1591  1 ! FUNCTIONAL DESCRIPTION:
;  1175          1592  1 !
;  1176          1593  1 !       This routine calls TPARSE to parse the given string with
;  1177          1594  1 !       the given state and key tables.
;  1178          1595  1 !
;  1179          1596  1 ! FORMAL PARAMETERS:
;  1180          1597  1 !
;  1181          1598  1 !       STRING    - Address of a string descriptor of string to parse
;  1182          1599  1 !       STATE_TAB - Address of TPARSE state tables
;  1183          1600  1 !       KEY_TAB   - Address of TPARSE key tables
;  1184          1601  1 !
;  1185          1602  1 ! IMPLICIT INPUTS:
;  1186          1603  1 !
;  1187          1604  1 !       None.
;  1188          1605  1 !
;  1189          1606  1 ! IMPLICIT OUTPUTS:
;  1190          1607  1 !
;  1191          1608  1 !       None.
;  1192          1609  1 !
;  1193          1610  1 ! ROUTINE VALUE:
;  1194          1611  1 ! COMPLETION CODES:
;  1195          1612  1 !
;  1196          1613  1 !       Returns completion code of LIB$TPARSE
;  1197          1614  1 !
;  1198          1615  1 ! SIDE EFFECTS:
;  1199          1616  1 !
;  1200          1617  1 !       None.
;  1201          1618  1 !--
;  1202          1619  1
;  1203          1620  2     BEGIN
;  1204          1621  2
;  1205          1622  2     LOCAL
;  1206          1623  2         TPARSE_BLOCK : BLOCK [TPA$K_LENGTH0, BYTE];
;  1207          1624  2
;  1208          1625  2     !
;  1209          1626  2     ! Initialize the TPARSE parameter block
;  1210          1627  2     !
;  1211          1628  2     TPARSE_BLOCK [TPA$L_COUNT]     = TPA$K_COUNT0;
;  1212          1629  2     TPARSE_BLOCK [TPA$L_OPTIONS]   = TPA$M_ABBREV;
;  1213          1630  2     TPARSE_BLOCK [TPA$L_STRINGCNT] = .STRING [STR$H_LENGTH];
;  1214          1631  2     TPARSE_BLOCK [TPA$L_STRINGPTR] = .STRING [STR$A_POINTER];
;  1215          1632  2     TPARSE_BLOCK [TPA$L_TOKENCNT]  = 0;
;  1216          1633  2     TPARSE_BLOCK [TPA$L_TOKENPTR]  = 0;
;  1217          1634  2     TPARSE_BLOCK [TPA$L_NUMBER]    = 0;
;  1218          1635  2     TPARSE_BLOCK [TPA$L_PARAM]     = 0;
;  1219          1636  2
;  1220          1637  2     !
;  1221          1638  2     ! Parse the string and return parse status
;  1222          1639  2     !
;  1223          1640  2     RETURN LIB$TPARSE (TPARSE_BLOCK, .STATE_TAB, .KEY_TAB);
;  1224          1641  1     END;
```

```
                                        0000 00000 CALL_TPARSE:
                                                           .WORD   Save nothing                                    ; 1588
                          5E                    20  C2 00002          SUBL2   #32, SP                                 ;
                                                08  DD 00005          PUSHL   #8                                      ; 1628
                    04    AE                    02  D0 00007          MOVL    #2, TPARSE_BLOCK+4                      ; 1629
                          50              04    AC  D0 00008          MOVL    STRING, R0                              ; 1630
                    08    AE                    60  3C 0000F          MOVZWL  (R0), TPARSE_BLOCK+8                    ;
                    0C    AE              04    A0  D0 00013          MOVL    4(R0), TPARSE_BLOCK+12                  ; 1631
                                          10    AE  7C 00018          CLRQ    TPARSE_BLOCK+16                         ; 1632
                                          1C    AE  7C 0001B          CLRQ    TPARSE_BLOCK+28                         ; 1634
                          7E              08    AC  7D 0001E          MOVQ    STATE_TAB, -(SP)                        ; 1640
                                          08    AE  9F 00022          PUSHAB  TPARSE_BLOCK                            ;
                00000000G 00                    03  FB 00025          CALLS   #3, LIB$TPARSE                          ;
                                                    04 0002C          RET                                             ; 1641

; Routine Size:  45 bytes,    Routine Base:  $CODE$ + 0306
```

```
;  1226          1642  1 %SBTTL 'ENTER_PAGE -- Action routine - enter value for /PAGE_NUMBERS'
;  1227          1643  1 ROUTINE ENTER_PAGE =
;  1228          1644  1 !++
;  1229          1645  1 !
;  1230          1646  1 ! FUNCTIONAL DESCRIPTION:
;  1231          1647  1 !
;  1232          1648  1 !       This routine is called as an action routine by TPARSE.
;  1233          1649  1 !
;  1234          1650  1 !       It save the parameter value passed by TPARSE
;  1235          1651  1 !
;  1236          1652  1 ! FORMAL PARAMETERS:
;  1237          1653  1 !
;  1238          1654  1 !       AP [TPA$L_PARAM] - TRUE if STANDARD page numbers, FALSE otherwise
;  1239          1655  1 !
;  1240          1656  1 ! IMPLICIT INPUTS:
;  1241          1657  1 !
;  1242          1658  1 !       None
;  1243          1659  1 !
;  1244          1660  1 ! IMPLICIT OUTPUTS:
;  1245          1661  1 !
;  1246          1662  1 !       CMDBLK [NDX$V_STANDARD_PAGE] - is set to parameter value
;  1247          1663  1 !
;  1248          1664  1 ! ROUTINE VALUE:
;  1249          1665  1 ! COMPLETION CODES:
;  1250          1666  1 !
;  1251          1667  1 !       TRUE
;  1252          1668  1 !
;  1253          1669  1 ! SIDE EFFECTS:
;  1254          1670  1 !
;  1255          1671  1 !       None
;  1256          1672  1 !
;  1257          1673  1 !--
;  1258          1674  1
;  1259          1675  2     BEGIN
;  1260          1676  2
;  1261          1677  2     BUILTIN
;  1262          1678  2         AP;
;  1263          1679  2
;  1264          1680  2     MAP
;  1265          1681  2         AP : REF BLOCK [, BYTE];
;  1266          1682  2
;  1267          1683  2     CMDBLK [NDX$V_STANDARD_PAGE] = .AP [TPA$L_PARAM];
;  1268          1684  2     RETURN TRUE;
;  1269          1685  1     END;


                                                0000 00000 ENTER_PAGE:
                                                           .WORD   Save nothing                                    ;  1643
00000000G  EF           01          05      20  AC  FD 00002       INSV    32(AP), #5, #1, CMDBLK                  ;  1683
                            50              01  D0  0000C         MOVL    #1, R0                                  ;  1684
                                            04  0000F         RET                                             ;  1685

; Routine Size:  16 bytes,    Routine Base:  $CODE$ + 0333
```

```
;  1271          1686  1 %SBTTL 'OPEN_ERROR - Handle File Open Errors'
;  1272          1687  1 GLOBAL ROUTINE OPEN_ERROR (FUNCTION_CODE, PRIMARY_CODE, SECONDARY_CODE, IOB : REF $XPO_IOB ()) =
;  1273          1688  1 !++
;  1274          1689  1 !
;  1275          1690  1 ! FUNCTIONAL DESCRIPTION:
;  1276          1691  1 !
;  1277          1692  1 !       This routine is called as an Action Routine to report file open errors
;  1278          1693  1 !
;  1279          1694  1 ! FORMAL PARAMETERS:
;  1280          1695  1 !
;  1281          1696  1 !       FUNCTION_CODE   - XPORT failure action routine function code
;  1282          1697  1 !       PRIMARY_CODE    - primary failure completion code
;  1283          1698  1 !       SECONDARY_CODE  - secondary failure completion code
;  1284          1699  1 !       IOB             - Address of file IOB
;  1285          1700  1 !
;  1286          1701  1 ! IMPLICIT INPUTS:
;  1287          1702  1 !
;  1288          1703  1 !       None
;  1289          1704  1 !
;  1290          1705  1 ! IMPLICIT OUTPUTS:
;  1291          1706  1 !
;  1292          1707  1 !       None
;  1293          1708  1 !
;  1294          1709  1 ! ROUTINE VALUE:
;  1295          1710  1 ! COMPLETION CODES:
;  1296          1711  1 !
;  1297          1712  1 !       Returns the value of PRIMARY_CODE if success is indicated.
;  1298          1713  1 !
;  1299          1714  1 ! SIDE EFFECTS:
;  1300          1715  1 !
;  1301          1716  1 !       Signals a fatal error terminating program execution if failure
;  1302          1717  1 !       is indicated by PRIMARY_CODE.
;  1303          1718  1 !--
;  1304          1719  1
;  1305          1720  2     BEGIN
;  1306          1721  2
;  1307          1722  2     BIND
;  1308          1723  2         FILE_SPEC = .IOB [IOB$A_FILE_SPEC] : $STR_DESCRIPTOR (),
;  1309          1724  2         RESULTANT =  IOB [IOB$T_RESULTANT] : $STR_DESCRIPTOR ();
;  1310          1725  2
;  1311          1726  2     LOCAL
;  1312          1727  2         FILE_NAME : REF $STR_DESCRIPTOR ();
;  1313          1728  2
;  1314          1729  2     !
;  1315          1730  2     ! Point to best file name
;  1316          1731  2     !
;  1317          1732  2     FILE_NAME = (IF    .RESULTANT [STR$H_LENGTH] NEQ 0
;  1318          1733  3                  THEN RESULTANT
;  1319          1734  3                  ELSE FILE_SPEC);
;  1320          1735  2
;  1321          1736  2     IF  NOT .PRIMARY_CODE
;  1322          1737  2     THEN
;  1323          1738  3         BEGIN
;  1324          1739  3         !
;  1325          1740  3         ! File was not opened
;  1326          1741  3         !
;  1327          1742  3
```

```
;  1328          1743  1          IF .IOB [IOB$V_INPUT]
;  1329          1744  1          THEN
;  1330          1745  1              SIGNAL_STOP (INDEX$_OPENIN, 1, .FILE_NAME,
;  1331          1746  1                           .IOB [IOB$G_COMP_CODE], 1, .IOB [IOB$G_2ND_CODE])
;  1332          1747  1          ELSE
;  1333          1748  1              SIGNAL_STOP (INDEX$_OPENOUT, 1, .FILE_NAME,
;  1334          1749  1                           .IOB [IOB$G_COMP_CODE], T, .IOB [IOB$G_2ND_CODE]);
;  1335          1750  1
;  1336          1751  1          END;
;  1337          1752  1
;  1338          1753  1      RETURN .PRIMARY_CODE;
;  1339          1754  1      END;


                                    0000 00000          .ENTRY  OPEN_ERROR, Save nothing                                ; 1687
                            50  10  AC  D0 00002          MOVL    IOB, R0                                                 ; 1723
                                1C  A0  B5 00006          TSTW    28(R0)                                                  ; 1732
                                    06  13 00009          BEQL    1$                                                      ;
                            51  1C  A0  9E 0000B          MOVAB   28(R0), FILE_NAME                                       ;
                                    04  11 0000F          BRB     2$                                                      ;
                            51  04  A0  D0 00011 1$:      MOVL    4(R0), FILE_NAME                                        ;
                            35  08  AC  E8 00015 2$:      BLBS    PRIMARY_CODE, 5$                                        ; 1736
                            16  2E  A0  E9 00019          BLBC    46(R0), 3$                                              ; 1743
                                00DC    C0  DD 0001D      PUSHL   220(R0)                                                 ; 1746
                                    01  DD 00021          PUSHL   #1                                                      ; 1745
                                00D8    C0  DD 00023      PUSHL   216(R0)                                                 ; 1746
                                        51  DD 00027      PUSHL   FILE_NAME                                               ; 1745
                                    01  DD 00029          PUSHL   #1                                                      ;
                                00000000G   8F  DD 0002B  PUSHL   #DSRINDEX$_OPENIN                                      ;
                                    14  11 00031          BRB     4$                                                      ;
                                00DC    C0  DD 00033 3$:  PUSHL   220(R0)                                                 ; 1749
                                    01  DD 00037          PUSHL   #1                                                      ; 1748
                                00D8    C0  DD 00039      PUSHL   216(R0)                                                 ; 1749
                                        51  DD 0003D      PUSHL   FILE_NAME                                               ; 1748
                                    01  DD 0003F          PUSHL   #1                                                      ;
                                00000000G   8F  DD 00041  PUSHL   #DSRINDEX$_OPENOUT                                     ;
                    00000000G   00          FB 00047 4$:  CALLS   #6, LIB$STOP                                            ;
                            50  08  AC  D0 0004E 5$:      MOVL    PRIMARY_CODE, R0                                        ; 1753
                                    04  00052          RET                                                             ; 1754

; Routine Size:  83 bytes,    Routine Base:  $CODE$ + 0343
```

```
;  1341        L 1755  1 %IF DSRPLUS
;  1342        U 1756  1 %THEN
;  1343        U 1757  1
;  1344        U 1758  1 %SBTTL 'ENTER_MERGE -- Action routine - enter page merging parameter'
;  1345        U 1759  1 ROUTINE ENTER_MERGE =
;  1346        U 1760  1 !++
;  1347        U 1761  1 !
;  1348        U 1762  1 ! FUNCTIONAL DESCRIPTION:
;  1349        U 1763  1 !
;  1350        U 1764  1 !       This routine is called as an action routine by TPARSE.
;  1351        U 1765  1 !
;  1352        U 1766  1 !       It saves the parameter passed by TPARSE.
;  1353        U 1767  1 !
;  1354        U 1768  1 ! FORMAL PARAMETERS:
;  1355        U 1769  1 !
;  1356        U 1770  1 !       AP [TPA$L_PARAM] - TRUE if MERGE adjacent pages, FALSE otherwise
;  1357        U 1771  1 !
;  1358        U 1772  1 ! IMPLICIT INPUTS:
;  1359        U 1773  1 !
;  1360        U 1774  1 !       None
;  1361        U 1775  1 !
;  1362        U 1776  1 ! IMPLICIT OUTPUTS:
;  1363        U 1777  1 !
;  1364        U 1778  1 !       CMDBLK [NDX$V_PAGE_MERGE] - value is stored here
;  1365        U 1779  1 !
;  1366        U 1780  1 ! ROUTINE VALUE:
;  1367        U 1781  1 ! COMPLETION CODES:
;  1368        U 1782  1 !
;  1369        U 1783  1 !       TRUE
;  1370        U 1784  1 !
;  1371        U 1785  1 ! SIDE EFFECTS:
;  1372        U 1786  1 !
;  1373        U 1787  1 !       None
;  1374        U 1788  1 !
;  1375        U 1789  1 !--
;  1376        U 1790  1
;  1377        U 1791  1     BEGIN
;  1378        U 1792  1
;  1379        U 1793  1     BUILTIN
;  1380        U 1794  1         AP;
;  1381        U 1795  1
;  1382        U 1796  1     MAP
;  1383        U 1797  1         AP : REF BLOCK [, BYTE];
;  1384        U 1798  1
;  1385        U 1799  1     CMDBLK [NDX$V_PAGE_MERGE] = .AP [TPA$L_PARAM];
;  1386        U 1800  1     RETURN TRUE;
;  1387        U 1801  1     END;
;  1388        U 1802  1
;  1389          1803  1 %FI
```

```
;  1391        L 1804  1 %IF DSRPLUS
;  1392        U 1805  1 %THEN
;  1393        U 1806  1
;  1394        U 1807  1 %SBTTL 'ENTER_LAYOUT -- Action routine - save value of /LAYOUT qualifier'
;  1395        U 1808  1 ROUTINE ENTER_LAYOUT =
;  1396        U 1809  1 !++
;  1397        U 1810  1 !
;  1398        U 1811  1 ! FUNCTIONAL DESCRIPTION:
;  1399        U 1812  1 !
;  1400        U 1813  1 !       This routine is called as an action routine by TPARSE.
;  1401        U 1814  1 !
;  1402        U 1815  1 !       It stores the parameter passed by TPARSE in the command
;  1403        U 1816  1 !       line information block.
;  1404        U 1817  1 !
;  1405        U 1818  1 ! FORMAL PARAMETERS:
;  1406        U 1819  1 !
;  1407        U 1820  1 !       AP [TPA$L_PARAM] - Layout value
;  1408        U 1821  1 !
;  1409        U 1822  1 ! IMPLICIT INPUTS:
;  1410        U 1823  1 !
;  1411        U 1824  1 !       None
;  1412        U 1825  1 !
;  1413        U 1826  1 ! IMPLICIT OUTPUTS:
;  1414        U 1827  1 !
;  1415        U 1828  1 !       CMDBLK [NDX$H_LAYOUT] - Value is stored here
;  1416        U 1829  1 !
;  1417        U 1830  1 ! ROUTINE VALUE:
;  1418        U 1831  1 ! COMPLETION CODES:
;  1419        U 1832  1 !
;  1420        U 1833  1 !       TRUE
;  1421        U 1834  1 !
;  1422        U 1835  1 ! SIDE EFFECTS:
;  1423        U 1836  1 !
;  1424        U 1837  1 !       None
;  1425        U 1838  1 !
;  1426        U 1839  1 !--
;  1427        U 1840  1
;  1428        U 1841  1     BEGIN
;  1429        U 1842  1
;  1430        U 1843  1     BUILTIN
;  1431        U 1844  1         AP;
;  1432        U 1845  1
;  1433        U 1846  1     MAP
;  1434        U 1847  1         AP : REF BLOCK [, BYTE];
;  1435        U 1848  1
;  1436        U 1849  1     CMDBLK [NDX$H_LAYOUT] = .AP [TPA$L_PARAM];
;  1437        U 1850  1     RETURN TRUE;
;  1438        U 1851  1     END;
;  1439        U 1852  1
;  1440          1853  1 %FI
```

```
;  1442        L 1854  1 %IF DSRPLUS
;  1443        U 1855  1 %THEN
;  1444        U 1856  1
;  1445        U 1857  1 %SBTTL 'ENTER_FORMAT -- Action routine - save value of /FORMAT qualifier'
;  1446        U 1858  1 ROUTINE ENTER_FORMAT =
;  1447        U 1859  1 !++
;  1448        U 1860  1 !
;  1449        U 1861  1 ! FUNCTIONAL DESCRIPTION:
;  1450        U 1862  1 !
;  1451        U 1863  1 !       This routine is called as an action routine by TPARSE.
;  1452        U 1864  1 !
;  1453        U 1865  1 !       It stores the parameter passed by TPARSE in the command line
;  1454        U 1866  1 !       information block.
;  1455        U 1867  1 !
;  1456        U 1868  1 !       If the format is TMS=E, it sets up the correct character size
;  1457        U 1869  1 !       vector in CHRSIZ.
;  1458        U 1870  1 !
;  1459        U 1871  1 !       If the format is TEX=filename, it copies filename to TEX_FILE_NAME
;  1460        U 1872  1 !
;  1461        U 1873  1 ! FORMAL PARAMETERS:
;  1462        U 1874  1 !
;  1463        U 1875  1 !       AP [TPA$L_PARAM]        - Format type value
;  1464        U 1876  1 !       AP [TPA$L_STRINGCNT]    - Length of filename for TEX=filename
;  1465        U 1877  1 !       AP [TPA$L_STRINGPTR]    - Pointer to filename for TEX=filename
;  1466        U 1878  1 !
;  1467        U 1879  1 ! IMPLICIT INPUTS:
;  1468        U 1880  1 !
;  1469        U 1881  1 !       CHRSZE                  - Address of TMS 'E' character size vector
;  1470        U 1882  1 !
;  1471        U 1883  1 ! IMPLICIT OUTPUTS:
;  1472        U 1884  1 !
;  1473        U 1885  1 !       CMDBLK [NDX$H_FORMAT]   - Value stored here
;  1474        U 1886  1 !       CHRSIZ                  - Points to TMS 'E' character size vector
;  1475        U 1887  1 !                                 for TMS=E, points to TEX character size
;  1476        U 1888  1 !                                 vector for TEX=filename.
;  1477        U 1889  1 !       TEX_FILE_NAME           - String descriptor of TEX filename.
;  1478        U 1890  1 !
;  1479        U 1891  1 ! ROUTINE VALUE:
;  1480        U 1892  1 ! COMPLETION CODES:
;  1481        U 1893  1 !
;  1482        U 1894  1 !       TRUE
;  1483        U 1895  1 !
;  1484        U 1896  1 ! SIDE EFFECTS:
;  1485        U 1897  1 !
;  1486        U 1898  1 !       None
;  1487        U 1899  1 !--
;  1488        U 1900  1     BEGIN
;  1489        U 1901  1
;  1490        U 1902  1     BUILTIN
;  1491        U 1903  1         AP;
;  1492        U 1904  1
;  1493        U 1905  1     MAP
;  1494        U 1906  1         AP : REF BLOCK [, BYTE];
;  1495        U 1907  1
;  1496        U 1908  1     CMDBLK [NDX$H_FORMAT] = .AP [TPA$L_PARAM];
;  1497        U 1909  1
;  1498        U 1910  1     SELECTONE .CMDBLK [NDX$H_FORMAT] OF
```

```
; 1499        U 1911  1          SET
; 1500        U 1912  1
; 1501        U 1913  1          [TMS11_E]:
; 1502        U 1914  1              CHRSIZ = CHRSZE;                    ! TMS 'E' character set
; 1503        U 1915  1
; 1504        U 1916  1          [TEX]:
; 1505        U 1917  1              BEGIN
; 1506        U 1918  1              CHRSIZ = TEX_CHAR_SIZES;            ! TEX character sizes
; 1507        U 1919  1
; 1508        U 1920  1              $STR_COPY (TARGET = TEX_FILE_NAME,
; 1509        U 1921  1                  STRING = (.AP [TPA$C_STRINGCNT], .AP [TPA$L_STRINGPTR]));
; 1510        U 1922  1
; 1511        U 1923  1              END;
; 1512        U 1924  1
; 1513        U 1925  1          [OTHERWISE]:                            ! Do nothing for DSR or TMS=A
; 1514        U 1926  1              ;
; 1515        U 1927  1
; 1516        U 1928  1          TES;
; 1517        U 1929  1
; 1518        U 1930  1      RETURN TRUE;
; 1519        U 1931  1      END;
; 1520        U 1932  1
; 1521          1933  1 %FI
```

```
;  1523          L 1934  1 %IF DSRPLUS
;  1524          U 1935  1 %THEN
;  1525          U 1936  1
;  1526          U 1937  1 %SBTTL 'ENTER_SORT -- Action routine - enter sort type'
;  1527          U 1938  1 ROUTINE ENTER_SORT =
;  1528          U 1939  1 !++
;  1529          U 1940  1 !
;  1530          U 1941  1 ! FUNCTIONAL DESCRIPTION:
;  1531          U 1942  1 !
;  1532          U 1943  1 !       This routine is called as an action routine by TPARSE.
;  1533          U 1944  1 !
;  1534          U 1945  1 !       The parameter passed by TPARSE is stored in the sort type variable.
;  1535          U 1946  1 !
;  1536          U 1947  1 ! FORMAL PARAMETERS:
;  1537          U 1948  1 !
;  1538          U 1949  1 !       AP [TPA$L_PARAM] - Sort type value (TRUE or FALSE)
;  1539          U 1950  1 !
;  1540          U 1951  1 ! IMPLICIT INPUTS:
;  1541          U 1952  1 !
;  1542          U 1953  1 !       None
;  1543          U 1954  1 !
;  1544          U 1955  1 ! IMPLICIT OUTPUTS:
;  1545          U 1956  1 !
;  1546          U 1957  1 !       CMDBLK [NDX$V_WORD_SORT] - value stored here
;  1547          U 1958  1 !
;  1548          U 1959  1 ! ROUTINE VALUE:
;  1549          U 1960  1 ! COMPLETION CODES:
;  1550          U 1961  1 !
;  1551          U 1962  1 !       TRUE
;  1552          U 1963  1 !
;  1553          U 1964  1 ! SIDE EFFECTS:
;  1554          U 1965  1 !
;  1555          U 1966  1 !       None
;  1556          U 1967  1 !
;  1557          U 1968  1 !--
;  1558          U 1969  1
;  1559          U 1970  1     BEGIN
;  1560          U 1971  1
;  1561          U 1972  1     BUILTIN
;  1562          U 1973  1         AP;
;  1563          U 1974  1
;  1564          U 1975  1     MAP
;  1565          U 1976  1         AP : REF BLOCK [, BYTE];
;  1566          U 1977  1
;  1567          U 1978  1     CMDBLK [NDX$V_WORD_SORT] = .AP [TPA$L_PARAM];
;  1568          U 1979  1     RETURN TRUE;
;  1569          U 1980  1     END;
;  1570          U 1981  1
;  1571            1982  1 %FI
```

```
; 1573        L 1983  1 %IF DSRPLUS
; 1574        U 1984  1 %THEN
; 1575        U 1985  1
; 1576        U 1986  1 %SBTTL 'ENTER_ALPHA -- Action routine - enter nonalpha sort value'
; 1577        U 1987  1 ROUTINE ENTER_ALPHA =
; 1578        U 1988  1 !++
; 1579        U 1989  1 !
; 1580        U 1990  1 ! FUNCTIONAL DESCRIPTION:
; 1581        U 1991  1 !
; 1582        U 1992  1 !       This routine is called as an action routine by TPARSE.
; 1583        U 1993  1 !
; 1584        U 1994  1 !       The parameter passed by TPARSE is stored as the nonalpha sort value.
; 1585        U 1995  1 !
; 1586        U 1996  1 ! FORMAL PARAMETERS:
; 1587        U 1997  1 !
; 1588        U 1998  1 !       AP [TPA$L_PARAM] - nonalpha sort value
; 1589        U 1999  1 !
; 1590        U 2000  1 ! IMPLICIT INPUTS:
; 1591        U 2001  1 !
; 1592        U 2002  1 !       None
; 1593        U 2003  1 !
; 1594        U 2004  1 ! IMPLICIT OUTPUTS:
; 1595        U 2005  1 !
; 1596        U 2006  1 !       CMDBLK [NDX$H_NONALPHA] - value is stored here
; 1597        U 2007  1 !
; 1598        U 2008  1 ! ROUTINE VALUE:
; 1599        U 2009  1 ! COMPLETION CODES:
; 1600        U 2010  1 !
; 1601        U 2011  1 !       TRUE
; 1602        U 2012  1 !
; 1603        U 2013  1 ! SIDE EFFECTS:
; 1604        U 2014  1 !
; 1605        U 2015  1 !       None
; 1606        U 2016  1 !
; 1607        U 2017  1 !--
; 1608        U 2018  1
; 1609        U 2019  1     BEGIN
; 1610        U 2020  1
; 1611        U 2021  1     BUILTIN
; 1612        U 2022  1         AP;
; 1613        U 2023  1
; 1614        U 2024  1     MAP
; 1615        U 2025  1         AP : REF BLOCK [, BYTE];
; 1616        U 2026  1
; 1617        U 2027  1     CMDBLK [NDX$H_NONALPHA] = .AP [TPA$L_PARAM];
; 1618        U 2028  1     RETURN TRUE;
; 1619        U 2029  1     END;
; 1620        U 2030  1
; 1621          2031  1 %FI
```

```
;  1623        L 2032  1 %IF DSRPLUS
;  1624        U 2033  1 %THEN
;  1625        U 2034  1
;  1626        U 2035  1 %SBTTL 'OPTIONS_FILE -- Process options file'
;  1627        U 2036  1 ROUTINE OPTIONS_FILE : NOVALUE =
;  1628        U 2037  1 !++
;  1629        U 2038  1 !
;  1630        U 2039  1 ! FUNCTIONAL DESCRIPTION:
;  1631        U 2040  1 !
;  1632        U 2041  1 !       Parse lines of an options file
;  1633        U 2042  1 !
;  1634        U 2043  1 ! FORMAL PARAMETERS:
;  1635        U 2044  1 !
;  1636        U 2045  1 !       None
;  1637        U 2046  1 !
;  1638        U 2047  1 ! IMPLICIT INPUTS:
;  1639        U 2048  1 !
;  1640        U 2049  1 !       CMDBLK [NDX$T_INPUT_FILE] - Options file name
;  1641        U 2050  1 !       NDXOPTION                 - Address of options file parse tables
;  1642        U 2051  1 !
;  1643        U 2052  1 ! IMPLICIT OUTPUTS:
;  1644        U 2053  1 !
;  1645        U 2054  1 !       CMDBLK [NDX$T_INPUT_FILE] - Input file name
;  1646        U 2055  1 !
;  1647        U 2056  1 ! ROUTINE VALUE:
;  1648        U 2057  1 ! COMPLETION CODES:
;  1649        U 2058  1 !
;  1650        U 2059  1 !       None
;  1651        U 2060  1 !
;  1652        U 2061  1 ! SIDE EFFECTS:
;  1653        U 2062  1 !
;  1654        U 2063  1 !       None
;  1655        U 2064  1 !
;  1656        U 2065  1 !--
;  1657        U 2066  1
;  1658        U 2067  1     BEGIN
;  1659        U 2068  1
;  1660        U 2069  1     LOCAL
;  1661        U 2070  1         OPTIOB : $XPO_IOB ();
;  1662        U 2071  1
;  1663        U 2072  1     $XPO_IOB_INIT (IOB = OPTIOB);
;  1664        U 2073  1     $XPO_OPEN (IOB = OPTIOB, FILE_SPEC = CMDBLK [NDX$T_INPUT_FILE],
;  1665        U 2074  1         DEFAULT = '.OPT', FAILURE = OPEN_ERROR);
;  1666        U 2075  1
;  1667        U 2076  1     WHILE $XPO_GET (IOB = OPTIOB) EQL XPO$_NORMAL DO
;  1668        U 2077  1         BEGIN
;  1669        U 2078  1
;  1670        U 2079  1         LOCAL
;  1671        U 2080  1             CH,
;  1672        U 2081  1             LEN,
;  1673        U 2082  1             PTR;
;  1674        U 2083  1
;  1675        U 2084  1         !
;  1676        U 2085  1         ! Strip comments from input line
;  1677        U 2086  1         !
;  1678        U 2087  1         PTR = CH$FIND_CH (.OPTIOB [IOB$H_STRING], CH$PTR (.OPTIOB [IOB$A_STRING]), %C'!');
;  1679        U 2088  1
```

```
;  1680        U 2089  1          IF  NOT CH$FAIL (.PTR)
;  1681        U 2090  1          THEN
;  1682        U 2091  1              !
;  1683        U 2092  1              ! Remove '!' and everything after it
;  1684        U 2093  1              !
;  1685        U 2094  1              LEN = CH$DIFF (.PTR, CH$PTR (.OPTIOB [IOB$A_STRING]))
;  1686        U 2095  1          ELSE
;  1687        U 2096  1              LEN = .OPTIOB [IOB$H_STRING];
;  1688        U 2097  1
;  1689        U 2098  1          !
;  1690        U 2099  1          ! Remove trailing whitespace
;  1691        U 2100  1          !
;  1692        U 2101  1          PTR = CH$PLUS (CH$PTR (.OPTIOB [IOB$A_STRING]), .LEN - 1);
;  1693        U 2102  1
;  1694        U 2103  1          DECR I FROM .LEN - 1 TO 0 DO
;  1695        U 2104  1              BEGIN
;  1696        U 2105  1              CH = CH$RCHAR (.PTR);
;  1697        U 2106  1              PTR = CH$PLUS (.PTR, -1);
;  1698        U 2107  1
;  1699        U 2108  1              IF  (.CH NEQ %C' ') AND (.CH NEQ TAB)
;  1700        U 2109  1              THEN
;  1701        U 2110  1                  EXITLOOP;
;  1702        U 2111  1
;  1703        U 2112  1              LEN = .I;
;  1704        U 2113  1              END;
;  1705        U 2114  1
;  1706        U 2115  1          IF  .LEN GTR 0
;  1707        U 2116  1          THEN
;  1708        U 2117  1              BEGIN
;  1709        U 2118  1              !
;  1710        U 2119  1              ! We have something to parse
;  1711        U 2120  1              !
;  1712        U 2121  1              $STR_COPY (TARGET = OPTIONS_STR,
;  1713        U 2122  1                  STRING = $STR_CONCAT ('OPTIONS ', (.LEN, .OPTIOB [IOB$A_STRING])));
;  1714        U 2123  1
;  1715        U 2124  1              IF  NOT CLI$DCL_PARSE (OPTIONS_STR, NDXOPTION)
;  1716        U 2125  1              THEN
;  1717        U 2126  1                  !
;  1718        U 2127  1                  ! Error parsing input line
;  1719        U 2128  1                  !
;  1720        U 2129  1                  SIGNAL_STOP (INDEX$_SYNTAX, 1, OPTIOB [IOB$T_STRING]);
;  1721        U 2130  1
;  1722        U 2131  1              !
;  1723        U 2132  1              ! Get input file name
;  1724        U 2133  1              !
;  1725        U 2134  1              CLI$GET_VALUE (%ASCID 'INPUT', CMDBLK [NDX$T_INPUT_FILE]);
;  1726        U 2135  1
;  1727        U 2136  1              IF  CLI$GET_VALUE (%ASCID'INPUT', VALUE_STR)
;  1728        U 2137  1              THEN
;  1729        U 2138  1                  !
;  1730        U 2139  1                  ! More than one input file specified.
;  1731        U 2140  1                  !
;  1732        U 2141  1                  SIGNAL (INDEX$_IGNORED, 1, VALUE_STR, INDEX$_NOLIST, 0,
;  1733        U 2142  1                          INDEX$_TEXT, 1, OPTIOB [IOB$T_STRING]);
;  1734        U 2143  1
;  1735        U 2144  1              !
;  1736        U 2145  1              ! Process /BOOK_IDENTIFIER
```

```
;  1737          U 2146  1                !
;  1738          U 2147  1                PARSE_BOOK ();
;  1739          U 2148  1
;  1740          U 2149  1                !
;  1741          U 2150  1                ! Finally, process the input file
;  1742          U 2151  1                !
;  1743          U 2152  1                NDXINP ();
;  1744          U 2153  1
;  1745          U 2154  1                CMDBLK [NDX$V_INPUT_CONCAT] = TRUE;          ! Next file concatenated to this one
;  1746          U 2155  1                END;
;  1747          U 2156  1
;  1748          U 2157  1            END;
;  1749          U 2158  1
;  1750          U 2159  1        $XPO_CLOSE (IOB = OPTIOB);
;  1751          U 2160  1        END;
;  1752          U 2161  1
;  1753            2162  1 %FI
```

```
;  1755        L 2163  1 %IF DSRPLUS
;  1756        U 2164  1 %THEN
;  1757        U 2165  1
;  1758        U 2166  1 %SBTTL 'PARSE_BOOK -- Parse /BOOK_IDENTIFIER qualifier'
;  1759        U 2167  1 ROUTINE PARSE_BOOK : NOVALUE =
;  1760        U 2168  1 !++
;  1761        U 2169  1 !
;  1762        U 2170  1 ! FUNCTIONAL DESCRIPTION:
;  1763        U 2171  1 !
;  1764        U 2172  1 !       This routine is called to process the /BOOK_IDENTIFIER qualifier
;  1765        U 2173  1 !
;  1766        U 2174  1 ! FORMAL PARAMETERS:
;  1767        U 2175  1 !
;  1768        U 2176  1 !       None
;  1769        U 2177  1 !
;  1770        U 2178  1 ! IMPLICIT INPUTS:
;  1771        U 2179  1 !
;  1772        U 2180  1 !       CMDBLK  - Command line information block
;  1773        U 2181  1 !
;  1774        U 2182  1 ! IMPLICIT OUTPUTS:
;  1775        U 2183  1 !
;  1776        U 2184  1 !       CMDBLK [NDX$T_MASTER_BOOK] - Set to book name if doing a master index
;  1777        U 2185  1 !
;  1778        U 2186  1 ! ROUTINE VALUE:
;  1779        U 2187  1 ! COMPLETION CODES:
;  1780        U 2188  1 !
;  1781        U 2189  1 !       None
;  1782        U 2190  1 !
;  1783        U 2191  1 ! SIDE EFFECTS:
;  1784        U 2192  1 !
;  1785        U 2193  1 !       None
;  1786        U 2194  1 !
;  1787        U 2195  1 !--
;  1788        U 2196  1
;  1789        U 2197  1     BEGIN
;  1790        U 2198  1
;  1791        U 2199  1     IF  .CMDBLK [NDX$V_MASTER]
;  1792        U 2200  1     THEN
;  1793        U 2201  1         BEGIN
;  1794        U 2202  1         !
;  1795        U 2203  1         ! Doing a master index
;  1796        U 2204  1         !
;  1797        U 2205  1
;  1798        U 2206  1         IF  CLI$PRESENT (%ASCID'BOOK_IDENTIFIER')
;  1799        U 2207  1         THEN
;  1800        U 2208  1             !
;  1801        U 2209  1             ! User specified a book name
;  1802        U 2210  1             !
;  1803        U 2211  1             CLI$GET_VALUE (%ASCID'BOOK_IDENTIFIER', CMDBLK [NDX$T_MASTER_BOOK])
;  1804        U 2212  1         ELSE
;  1805        U 2213  1             BEGIN
;  1806        U 2214  1             !
;  1807        U 2215  1             ! Doing a master index and no book identifier specified.
;  1808        U 2216  1             ! Use input file name.
;  1809        U 2217  1             !
;  1810        U 2218  1             LOCAL
;  1811        U 2219  1                 PARSE_SPEC_BLOCK : $XPO_SPEC_BLOCK;
```

```
;  1812        U 2220  1
;  1813        U 2221  1              IF $XPO_PARSE_SPEC (FILE_SPEC = CMDBLK [NDX$T_INPUT_FILE],
;  1814        U 2222  1                   SPEC_BLOCK = PARSE_SPEC_BLOCK, FAILURE = 0)
;  1815        U 2223  1              THEN
;  1816        U 2224  1                  !
;  1817        U 2225  1                  ! Filename parse succeeded. Use filename as book name.
;  1818        U 2226  1                  !
;  1819        U 2227  1                  $STR_COPY (STRING = PARSE_SPEC_BLOCK [XPO$T_FILE_NAME],
;  1820        U 2228  1                       TARGET = CMDBLK [NDX$T_MASTER_BOOK])
;  1821        U 2229  1              ELSE
;  1822        U 2230  1                  !
;  1823        U 2231  1                  ! Filename parse failed. Use NULL book name.
;  1824        U 2232  1                  !
;  1825        U 2233  1                  $STR_COPY (STRING = '', TARGET = CMDBLK [NDX$T_MASTER_BOOK]);
;  1826        U 2234  1
;  1827        U 2235  1              END;
;  1828        U 2236  1          END
;  1829        U 2237  1      ELSE
;  1830        U 2238  1          !
;  1831        U 2239  1          ! Not doing a master index
;  1832        U 2240  1          !
;  1833        U 2241  1
;  1834        U 2242  1          IF  CLI$PRESENT (%ASCID'BOOK_IDENTIFIER')
;  1835        U 2243  1          THEN
;  1836        U 2244  1              SIGNAL (INDEX$_IGNORED, 1, %ASCID'BOOK_IDENTIFIER', INDEX$_CONFQUAL);
;  1837        U 2245  1
;  1838        U 2246  1      END;
;  1839        U 2247  1
;  1840          2248  1 %FI
```

```
;  1842        L 2249  1 %IF DSRPLUS
;  1843        U 2250  1 %THEN
;  1844        U 2251  1
;  1845        U 2252  1 %SBTTL 'PROCESS_TEX_FILE - Process TEX character size file'
;  1846        U 2253  1 ROUTINE PROCESS_TEX_FILE : NOVALUE =
;  1847        U 2254  1 !++
;  1848        U 2255  1 !
;  1849        U 2256  1 ! FUNCTIONAL DESCRIPTION:
;  1850        U 2257  1 !
;  1851        U 2258  1 !       This routine is called to process the TEX character size file.
;  1852        U 2259  1 !
;  1853        U 2260  1 ! FORMAL PARAMETERS:
;  1854        U 2261  1 !
;  1855        U 2262  1 !       None
;  1856        U 2263  1 !
;  1857        U 2264  1 ! IMPLICIT INPUTS:
;  1858        U 2265  1 !
;  1859        U 2266  1 !       TEX_FILE_NAME   - String descriptor of TEX character size file name
;  1860        U 2267  1 !
;  1861        U 2268  1 ! IMPLICIT OUTPUTS:
;  1862        U 2269  1 !
;  1863        U 2270  1 !       TEX_FILE_NAME   - Replaced with best file name during file processing.
;  1864        U 2271  1 !       TEX_CHAR_INDEX  - Initialized to zero
;  1865        U 2272  1 !       TEX_CHAR_SIZES  - Initialized to zero
;  1866        U 2273  1 !       TEX_FILE_LINE_NO- Initialized to one
;  1867        U 2274  1 !
;  1868        U 2275  1 ! ROUTINE VALUE:
;  1869        U 2276  1 ! COMPLETION CODES:
;  1870        U 2277  1 !
;  1871        U 2278  1 !       None
;  1872        U 2279  1 !
;  1873        U 2280  1 ! SIDE EFFECTS:
;  1874        U 2281  1 !
;  1875        U 2282  1 !       None
;  1876        U 2283  1 !--
;  1877        U 2284  1     BEGIN
;  1878        U 2285  1     TEX_FILE_LINE_NO = 1;
;  1879        U 2286  1
;  1880        U 2287  1     TEX_CHAR_INDEX = 0;
;  1881        U 2288  1     INCR I FROM 0 TO 255 DO TEX_CHAR_SIZES [.I] =0;
;  1882        U 2289  1
;  1883        U 2290  1     !
;  1884        U 2291  1     ! Set filename and open the file
;  1885        U 2292  1     !
;  1886        U 2293  1     TEX_FAB [FAB$B_FNS] = .TEX_FILE_NAME [STR$H_LENGTH];
;  1887        U 2294  1     TEX_FAB [FAB$L_FNA] = .TEX_FILE_NAME [STR$A_POINTER];
;  1888        U 2295  1     $OPEN (FAB = TEX_FAB);
;  1889        U 2296  1     !
;  1890        U 2297  1     ! Get the best file name
;  1891        U 2298  1     !
;  1892        U 2299  1     IF .TEX_NAM [NAM$B_RSL] NEQ 0
;  1893        U 2300  1     THEN                                        ! Use resultant name
;  1894        U 2301  1         $STR_COPY (TARGET = TEX_FILE_NAME,
;  1895        U 2302  1             STRING = (.TEX_NAM [NAM$B_RSL], .TEX_NAM [NAM$L_RSA]))
;  1896        U 2303  1     ELSE                                        ! No resultant name
;  1897        U 2304  1         BEGIN
;  1898        U 2305  1
```

```
; 1899        U 2306  1          IF .TEX_NAM [NAM$B_ESL] NEQ 0
; 1900        U 2307  1          THEN                                            ! Use expanded name
; 1901        U 2308  1              $STR_COPY (TARGET = TEX_FILE_NAME,
; 1902        U 2309  1                  STRING = (.TEX_NAM [NAM$B_ESL], .TEX_NAM [NAM$L_ESA]));
; 1903        U 2310  1
; 1904        U 2311  1          END;
; 1905        U 2312  1
; 1906        U 2313  1      IF NOT .TEX_FAB [FAB$L_STS]
; 1907        U 2314  1      THEN                                                ! File not open
; 1908        U 2315  1          SIGNAL_STOP (INDEX$_OPENIN, 1, TEX_FILE_NAME,
; 1909        U 2316  1              .TEX_FAB [FAB$L_STS], .TEX_FAB [FAB$L_STV]);
; 1910        U 2317  1
; 1911        U 2318  1      IF NOT $CONNECT (RAB = TEX_RAB)                     ! Connect record stream
; 1912        U 2319  1      THEN                                                !  - error
; 1913        U 2320  1          SIGNAL_STOP (INDEX$_OPENIN, 1, TEX_FILE_NAME,
; 1914        U 2321  1              .TEX_RAB [RAB$L_STS], .TEX_RAB [RAB$L_STV]);
; 1915        U 2322  1
; 1916        U 2323  1      $GET (RAB = TEX_RAB);                               ! Get first line in file
; 1917        U 2324  1      $STR_DESC_INIT (DESCRIPTOR = TEX_LINE,
; 1918        U 2325  1          STRING = (.TEX_RAB [RAB$W_RSZ], .TEX_RAB [RAB$L_RBF]));
; 1919        U 2326  1
; 1920        U 2327  1      IF .TEX_RAB [RAB$L_STS]
; 1921        U 2328  1      THEN
; 1922        U 2329  1          BEGIN
; 1923        U 2330  1          !
; 1924        U 2331  1          ! Process TEX character size file
; 1925        U 2332  1          !
; 1926        U 2333  1          IF RMS$_EOF NEQ CALL_TPARSE (TEX_LINE, TEX_FILE_STATE, TEX_FILE_KEY)
; 1927        U 2334  1          THEN
; 1928        U 2335  1              SIGNAL_STOP (INDEX$_TEXFILE, 2, .TEX_FILE_LINE_NO, TEX_FILE_NAME,
; 1929        U 2336  1                  INDEX$_SYNTAX, T, TEX_LINE);
; 1930        U 2337  1
; 1931        U 2338  1          END;
; 1932        U 2339  1
; 1933        U 2340  1      IF .TEX_CHAR_INDEX LSS 128
; 1934        U 2341  1      THEN                                                ! Not enough values supplied
; 1935        U 2342  1          SIGNAL (INDEX$_TEXFILE, 2, .TEX_FILE_LINE_NO, TEX_FILE_NAME, INDEX$_TOOFEW);
; 1936        U 2343  1
; 1937        U 2344  1      $CLOSE (FAB = TEX_FAB);
; 1938        U 2345  1      END;
; 1939        U 2346  1
; 1940          2347  1  %FI
```

```
;  1942          L 2348  1 %IF DSRPLUS
;  1943          U 2349  1 %THEN
;  1944          U 2350  1
;  1945          U 2351  1 %SBTTL 'STORE_TEX - Action routine - Store TEX character size'
;  1946          U 2352  1 ROUTINE STORE_TEX =
;  1947          U 2353  1 !++
;  1948          U 2354  1 !
;  1949          U 2355  1 ! FUNCTIONAL DESCRIPTION:
;  1950          U 2356  1 !
;  1951          U 2357  1 !       This routine is called as an action routine by LIB$TPARSE to
;  1952          U 2358  1 !       store a TEX character size.
;  1953          U 2359  1 !
;  1954          U 2360  1 ! FORMAL PARAMETERS:
;  1955          U 2361  1 !
;  1956          U 2362  1 !       AP [TPA$L_NUMBER]       - Value to be stored
;  1957          U 2363  1 !
;  1958          U 2364  1 ! IMPLICIT INPUTS:
;  1959          U 2365  1 !
;  1960          U 2366  1 !       TEX_CHAR_INDEX          - Index into TEX_CHAR_SIZES where next value
;  1961          U 2367  1 !                                 is to be stored
;  1962          U 2368  1 !
;  1963          U 2369  1 ! IMPLICIT OUTPUTS:
;  1964          U 2370  1 !
;  1965          U 2371  1 !       TEX_CHAR_SIZES [.TEX_CHAR_INDEX]- Contains value
;  1966          U 2372  1 !       TEX_CHAR_INDEX                  - Is incremented
;  1967          U 2373  1 !
;  1968          U 2374  1 ! ROUTINE VALUE:
;  1969          U 2375  1 ! COMPLETION CODES:
;  1970          U 2376  1 !
;  1971          U 2377  1 !       TRUE
;  1972          U 2378  1 !
;  1973          U 2379  1 ! SIDE EFFECTS:
;  1974          U 2380  1 !
;  1975          U 2381  1 !       Signals a fatal error if TEX_CHAR_INDEX exceeds 255
;  1976          U 2382  1 !--
;  1977          U 2383  1     BEGIN
;  1978          U 2384  1
;  1979          U 2385  1     BUILTIN
;  1980          U 2386  1         AP;
;  1981          U 2387  1
;  1982          U 2388  1     MAP
;  1983          U 2389  1         AP : REF BLOCK [, BYTE];
;  1984          U 2390  1
;  1985          U 2391  1     IF .TEX_CHAR_INDEX EQL 256
;  1986          U 2392  1     THEN
;  1987          U 2393  1         SIGNAL_STOP (INDEX$_TEXFILE, 2, .TEX_FILE_LINE_NO, TEX_FILE_NAME, INDEX$_TOOMANY);
;  1988          U 2394  1
;  1989          U 2395  1     TEX_CHAR_SIZES [.TEX_CHAR_INDEX] = .AP [TPA$L_NUMBER];
;  1990          U 2396  1     TEX_CHAR_INDEX = .TEX_CHAR_INDEX + 1;
;  1991          U 2397  1     RETURN TRUE;
;  1992          U 2398  1     END;
;  1993          U 2399  1
;  1994            2400  1 %FI
```

```
;  1996        L 2401  1 %IF DSRPLUS
;  1997        U 2402  1 %THEN
;  1998        U 2403  1
;  1999        U 2404  1 %SBTTL 'READ_TEX -- Action routine - Read a record from TEX char size file'
;  2000        U 2405  1 ROUTINE READ_TEX =
;  2001        U 2406  1 !++
;  2002        U 2407  1 !
;  2003        U 2408  1 ! FUNCTIONAL DESCRIPTION:
;  2004        U 2409  1 !
;  2005        U 2410  1 !       This routine is called as an action routine by TPARSE.
;  2006        U 2411  1 !
;  2007        U 2412  1 !       It reads a line from the input file.
;  2008        U 2413  1 !
;  2009        U 2414  1 ! FORMAL PARAMETERS:
;  2010        U 2415  1 !
;  2011        U 2416  1 !       None
;  2012        U 2417  1 !
;  2013        U 2418  1 ! IMPLICIT INPUTS:
;  2014        U 2419  1 !
;  2015        U 2420  1 !       TEX_RAB                 - RMS RAB to read
;  2016        U 2421  1 !
;  2017        U 2422  1 ! IMPLICIT OUTPUTS:
;  2018        U 2423  1 !
;  2019        U 2424  1 !       TEX_IN_BUF              - Contains text of new line
;  2020        U 2425  1 !       TEX_LINE                - Is a string descriptor of new line
;  2021        U 2426  1 !       TEX_FILE_LINE_NO        - Is incremented
;  2022        U 2427  1 !       AP [TPA$L_STRINGCNT]    - Is length of new line
;  2023        U 2428  1 !       AP [TPA$L_STRINGPTR]    - Points to new line
;  2024        U 2429  1 !
;  2025        U 2430  1 ! ROUTINE VALUE:
;  2026        U 2431  1 ! COMPLETION CODES:
;  2027        U 2432  1 !
;  2028        U 2433  1 !       Returns TRUE if successful
;  2029        U 2434  1 !       Returns RMS$_EOF if end of file encountered
;  2030        U 2435  1 !       Returns FALSE otherwise
;  2031        U 2436  1 !
;  2032        U 2437  1 ! SIDE EFFECTS:
;  2033        U 2438  1 !
;  2034        U 2439  1 !       None
;  2035        U 2440  1 !--
;  2036        U 2441  1     BEGIN
;  2037        U 2442  1     BUILTIN
;  2038        U 2443  1         AP;
;  2039        U 2444  1
;  2040        U 2445  1     MAP
;  2041        U 2446  1         AP : REF BLOCK [, BYTE];
;  2042        U 2447  1
;  2043        U 2448  1     IF NOT $GET (RAB = TEX_RAB)
;  2044        U 2449  1     THEN
;  2045        U 2450  1         RETURN (IF .TEX_RAB [RAB$L_STS] EQL RMS$_EOF THEN RMS$_EOF ELSE FALSE);
;  2046        U 2451  1
;  2047        U 2452  1     TEX_FILE_LINE_NO = .TEX_FILE_LINE_NO + 1;
;  2048        U 2453  1     $STR_DESC_INIT (DESCRIPTOR = TEX_LINE,
;  2049        U 2454  1         STRING = (.TEX_RAB [RAB$W_RSZ], .TEX_RAB [RAB$L_RBF]));
;  2050        U 2455  1
;  2051        U 2456  1     AP [TPA$L_STRINGCNT] = .TEX_LINE [STR$H_LENGTH];
;  2052        U 2457  1     AP [TPA$L_STRINGPTR] = .TEX_LINE [STR$A_POINTER];
```

```
; 2053        U 2458  1      RETURN TRUE;
; 2054        U 2459  1      END;
; 2055        U 2460  1
; 2056          2461  1 %FI
; 2057          2462  1
; 2058          2463  1 END                          ! End of module
; 2059          2464  0 ELUDOM



                                                                .EXTRN  LIB$SIGNAL, LIB$STOP

;                                PSECT SUMMARY
;
;       Name                    Bytes                           Attributes
;
;  $OWN$                            24  NOVEC,  WRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
;  _LIB$KEY0$                        4  NOVEC,NOWRT,  RD ,  EXE,  SHR,  LCL,  REL,  CON,  PIC,ALIGN(1)
;  _LIB$STATE$                      42  NOVEC,NOWRT,  RD ,  EXE,  SHR,  LCL,  REL,  CON,  PIC,ALIGN(1)
;  _LIB$KEY1$                       19  NOVEC,NOWRT,  RD ,  EXE,  SHR,  LCL,  REL,  CON,  PIC,ALIGN(1)
;  $PLIT$                          276  NOVEC,NOWRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
;  $CODE$                          918  NOVEC,NOWRT,  RD ,  EXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)



;                          Library Statistics
;
;                                    --------- Symbols ---------      Pages       Processing
;       File                         Total    Loaded   Percent      Mapped       Time
;
;  _$255$DUA28:[SYSLIB]STARLET.L32;1    9776        20         0         581       00:01.0
;  _$255$DUA28:[SYSLIB]TPAMAC.L32;1       42        25        59          14       00:00.1
;  _$255$DUA28:[SYSLIB]XPORT.L32;1       590       119        20         252       00:00.6



;                                  COMMAND QUALIFIERS

;       BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/LIS=LIS$:NDXVMS/OBJ=OBJ$:NDXVMS MSRC$:NDXVMS/UPDATE=(ENH$:NDXVMS)

; Size:         918 code + 365 data bytes
; Run Time:         00:42.1
; Elapsed Time:     01:22.5
; Lines/CPU Min:     3514
; Lexemes/CPU-Min: 45469
; Memory Used:  273 pages
; Compilation Complete
```

NDXTEX
LIS

NDXTMS
LIS

NDXPAG
LIS

NDXVMS
LIS

NDXVMSMSG
LIS